ISTEC | INSTITUTO SUPERIOR DE TECNOLOGIAS AVANÇADAS

19

Projeto Global

Licenciatura em Informática

Aluno: Alexandre Miguel Gomes Lima
Nº: 8142
Orientador: Pedro Brandão

Agradecimentos

Gostaria de agradecer a todos os que me apoiaram no desenvolvimento deste percurso e na realização deste projeto, mais concretamente a minha mãe, o meu pai e a minha companheira pelos sacrifícios que fizeram para me permitirem chegar a este patamar. Não esquecendo todos os professores que me ajudaram a atingir o nível pretendido e pela disponibilidade de cada um para esclarecer dúvidas que surgiram, mais especificamente o professor José Neves, o professor Pedro Crispim e ao doutor Pedro Brandão. Por fim gostaria de agradecer aos meus colegas que me incentivaram sempre a querer mais e um especial obrigado aos alunos Tiago Costa e Hugo Amorim por me acompanharem nesta jornada sempre com grande companheirismo.

Resumo

Este trabalho tem como objetivo simular um ambiente empresarial, para tal, através da virtualização de vários servidores obteve-se uma infraestrutura apta para aceder e desenvolver uma aplicação WPF através da rede. Esta aplicação irá simular uma biblioteca onde os membros do domínio poderão acedê-la conforme as suas permissões.

Desta forma, é necessário compreender as tecnologias que permitem a sua implementação. A base de todo o projeto assenta na virtualização, que é a responsável por permitir que uma máquina física dê origem a múltiplas máquinas com diferentes funções e sistemas operativos. As restantes tecnologias como o sistema operativo Windows Server 2012 R2, o servidor SQL ou o Visual Studio foram resultados da aprendizagem ao longo do curso. Os resultados obtidos através da alta disponibilidade, criada pela estrutura, permitiram as condições necessárias para o desenvolvimento de uma aplicação capaz de gerir o sistema informático de uma biblioteca.

Em síntese, os resultados demonstram as capacidades das tecnologias utilizadas, ajudando a perceber as vantagens que oferecem e como poderiam ser equacionadas em empresas, tanto na renovação ou criação das suas infraestruturas como na criação de sistemas informáticos.

Palavras-chave: Virtualização – WPF – Infraestrutura – Windows Server 2012

Abstract

This work aims to simulate a business environment, through the virtualization of multiple servers creating an infrastructure capable of access and develop a WPF application over the network. This application will simulate a library where members of the domain can access it according to their permissions.

Thus, it is necessary to understand the technologies that allow its implementation. The concepts that will be addressed will clarify the reader in relation to each one of them. The basis of the whole project is based on virtualization, which is responsible for allowing a physical machine to originate multiple machines with different functions and operating systems. The remaining technologies like the Windows Server 2012 R2 operating system or the SQL server were a result of learning along the course. The results obtained through high availability, generated by the structure, allowed the necessary conditions for the development of an application capable of managing the computer system of a library.

In summary, the results show the capabilities of the technologies used, helping to realize the advantages they offer and how they could be solved in companies, both in the renovation or creation of their infrastructures and in the creation of computer systems.

Keywords: Virtualization - WPF - Infrastructure - Windows Server 2012

Abreviaturas

AD	Active Directory
ADDS	Active Directory Domain Services
CSV	Cluster Shared Volume
DC	Domain Controller
DNS	Domain Name System
IP	Internet Protocol
LINQ	Language Integrated Query
SQL	Structured Query Language
WPF	Windows Presentation Foundation

Índice

Índice de Figuras

Índice de Tabelas

Introdução

Este projeto tem como objetivo demonstrar as capacidades que a virtualização oferece bem como a criação de uma aplicação WPF num ambiente virtualizado, responsável pela gestão de uma biblioteca através da rede de domínio.

Desta forma, com a virtualização de vários servidores, tais como, um *Domain Controler*, uma máquina de *Router*, duas máquinas de servidor SQL, uma máquina de *Storage* e uma máquina de Cliente (*Desktop*) irá ser montada uma infraestrutura capaz de simular um ambiente empresarial com alta disponibilidade.

Ao longo do trabalho irão ser abordados diferentes temas. Através do estado de arte pretende-se corroborar os conhecimentos necessários para a criação do laboratório, sendo as tecnologias como a virtualização, o sistema operativo Windows Server 2012 R2, o Active Directory ou a *framework .NET* abordadas. No capítulo de desenvolvimento será explicado como foi criada e configurada a infraestrutura e como foi desenvolvida a aplicação juntamente com a sua base de dados.

Em suma, pretende-se simular o funcionamento de uma biblioteca, desde os servidores necessários para a criação de uma infraestrutura sustentável até à aplicação que interage com os seus funcionários. Assim sendo, através da virtualização consegue-se criar uma rede de gestão empresarial capaz de fazer os objetivos propostos.

Estado de Arte

Virtualização

A virtualização surgiu na época dos *mainframes*, por volta dos anos sessenta com o objetivo de aumentar a eficiência das unidades primárias de processamento para vários grupos. Neste período, a IBM lançou o primeiro *mainframe* que era capaz de executar vários sistemas de uma só vez. Nos anos oitenta e na década seguinte houve um desinvestimento desta tecnologia com o aparecimento da arquitetura x86 e dos computadores *Desktop*. (Marcelo Cortêz, 2013, p. 2)

A utilização desta arquitetura apenas permitia uma utilização de 10% a 15% da sua potencialidade. No início do novo século, uma empresa chamada VMware descobriu uma forma de virtualizar máquinas com base x86. Posteriormente, a Intel e a AMD adicionaram aos seus processadores a capacidade de serem virtualizados o que permitiu maiores avanços de forma mais facilitada. Com os efeitos benéficos que esta tecnologia traz, tem então crescido exponencialmente até aos dias de hoje. (Talaber, 2009, p. 10)

Pode-se definir virtualização como uma abstração de um componente físico num objeto lógico (Portnoy, 2016, p. 2). Esta tecnologia permite separar o *software* do *hardware* como por exemplo sistemas operativos (SO) ou aplicações, assim passou a ser possível utilizar vários SO em uma máquina de forma isolada, partilhando os seus recursos físicos por todos. Por outras palavras trata-se de criar um ambiente virtual na máquina física onde podem existir vários sistemas operativos isolados. Desta maneira tornou-se vantajoso utilizar a virtualização de várias maneiras para obter resultados superiores que não seriam possíveis sem ela. (*ibidem*, p. 26)

Ao virtualizar-se vários SO numa só máquina de forma isolada obtém-se várias instâncias de SO, denominadas de Máquinas Virtuais (MV). O *software* responsável por fazer a ligação entre os recursos físicos e as MV bem como a sua interface é chamado de *hypervisor*. Geralmente o *hypervisor* é implementado por *software* mas também o pode ser feito por *firmware*. Se uma máquina tivesse várias MV diretamente sobre o *hardware*, sem ter um *hypervisor* no meio, todas elas iriam querer os recursos de forma exclusiva, assim o trabalho do *hypervisor* é gerir cada máquina virtual bem como a partilha de recursos do *Host*. (Tulloch, 2010, p. 23)

A camada onde o *hypervisor* se encontra depende da maneira como é instalado, conforme se pode visualizar na figura 1. Se for instalado sobre o *hardware* sem nenhum tipo de sistema operativo é chamado de *bare metal* ou de virtualização nativa. É então considerado do tipo 1 e utiliza-se mais a nível *datacenters* visto que não existe desperdício de recursos da máquina uma vez que o *hypervisor* está a fazer a alocação de recursos diretamente entre o *hardware* e as MV. O *hypervisor* de tipo 2 é instalado num sistema operativo, desta forma o desempenho é inferior comparativamente com os do tipo 1 porque existe uma camada entre o *hardware* e o *hypervisor*. Assim quando se faz qualquer operação numa MV, o *hypervisor* comunica com o sistema operativo em vez de comunicar diretamente com o *hardware*. Este tipo também é mais suscetível a falhas ou interrupções, uma vez que a falha do sistema operativo *Host* irá afetar todas as VM. Mas em termos de compatibilidade o *hypervisor* de tipo 2 tem um alcance maior, visto que a maioria das configurações já foi feita pelo SO. (Portnoy, 2016, pp. 23-26)

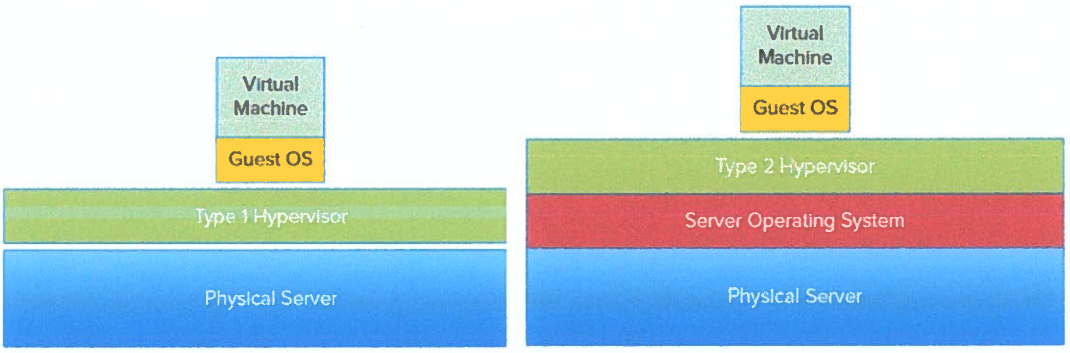

Figura 1- Tipos de Hypervisor
(Portnoy, 2016)

Além de controlar o acesso aos recursos pelas MV, esta tecnologia através de várias empresas como a Microsoft, com o *Hyper-V,* consegue migrar máquinas virtuais de um servidor físico para outro (*Live Migration*), replicá-los de forma a manter a sua infraestrutura sempre disponível em caso de falhas (*Hyper-V Replica*) ou atribuir memória dinâmica de forma eficiente. (Microsoft, 2016)

Define-se uma máquina virtual como a virtualização dos recursos de *hardware*, incluindo o processador, memória ROM e RAM ou rede. Estes recursos que a MV está a aceder não existem em concreto, uma vez que o *hipervisor* faz uma abstração dos recursos físicos de forma a serem obtidos por *software*. Assim todos os dispositivos como discos, processadores ou interfaces de rede na realidade são dispositivos virtuais padronizados para cada uma das máquinas. Pode-se realizar várias operações nas MV tais como clones, *snapshots*, *backups* ou *templates*. (Portnoy, 2016, pp. 37-39).

A figura abaixo, representa uma máquina virtual no Hyper-V.

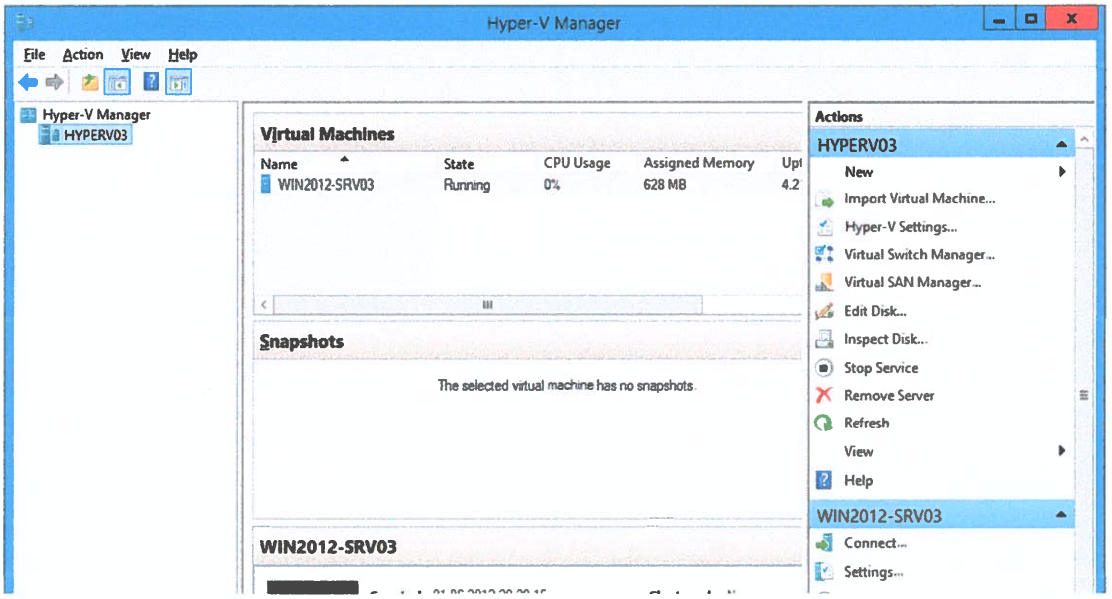

Figura 2- Máquina Virtual
(Microsoft, 2013)

Tipos de virtualização

A virtualização pode ser entendida como uma maneira de centralizar a gestão, mas principalmente diminuir drasticamente os custos de equipamentos físicos desde servidores a computadores como até mesmo de *routers* ou *switches*, não esquecendo as manutenções ou despesas, desta forma existem vários tipos de virtualização. A virtualização de servidores consiste na virtualização de vários servidores individuais que são executados sobre o mesmo *hardware* desta forma obtêm-se uma maior disponibilidade e escalabilidade, uma vez que a probabilidade de haver falhas físicas irá ser menor comparativamente a ter vários servidores físicos e porque rapidamente se consegue aumentar o número de servidores da arquitetura. (Tulloch, 2010, pp. 21-22)

A virtualização de *Desktops* assenta no mesmo conceito que o anterior, assim o servidor que está no *datacenter* vai virtualizar *Desktops* em vez de servidores. Estes computadores virtuais são então acedidos pelos utilizadores através de uma aplicação ou de um *browser*. Assim atinge-se melhor desempenho visto que o servidor físico tem mais capacidades do que os Desktops habituais. Desta maneira, consegue-se uma gestão de forma mais simplificada visto que não irá haver tantas falhas físicas. (*ibidem*, p.109) A virtualização de *storage* tem algumas semelhanças com o conceito de *RAID* mas define-se pelos de *cluster* e de *storage* partilhado. Desta forma agrupam-se vários discos físicos num grupo ou *pool*. Estes discos podem estar em diferentes localizações desde que exista acesso por rede (Microsoft, 2013). Quando se tem várias MV num só *Host* e se pretende que estas comuniquem umas com as outras, esta ligação só é possível graças ao *hypervisor* que irá criar uma rede interna através da virtualização de rede. Assim, tal como um *switch* físico, irá existir um *switch* virtual na máquina *Host* que irá isolar o tráfego para a rede interna e outro *switch* para a rede externa. (Portnoy, 2016, pp. 179-180)

A virtualização abriu as portas ao avanço tecnológico quando a lei de Moore fazia prever que se estava a chegar ao limiar das capacidades computacionais (*ibidem*, p. 9). Esta tecnologia traz várias vantagens como a performance, custos e uma rápida implementação. A nível de desempenho visto que através da abstração de recursos consegue-se ter várias VM a utilizar os recursos que apenas eram utilizados por um SO. Em relação aos custos a

virtualização é utilizada com o objetivo de reduzir os custos de aquisição de equipamentos, as despesas de utilização como eletricidade, arrefecimento ou falhas de *hardware*. A nível de implementação rapidamente se consegue a reposição de um servidor em caso de falha em relação a um servidor físico, com a utilização de um *snapshot* ou de um *backup*. Além destas vantagens também se junta a flexibilidade e a escalabilidade devido à velocidade com que se cria e configura uma MV comparativamente às máquinas físicas. Uma das desvantagens que a virtualização apresenta é o fato de ser possível atribuir recursos que não existem. Por exemplo, se três MV tiverem 10 GB de armazenamento em modo dinâmico ou fixo e o *Host* apenas possuir 20 GB, então se as três máquinas precisarem ao mesmo tempo de memória haverá problemas (se for em modo dinâmico, caso fosse em modo fixo bastava estarem ligadas) uma vez que a máquina física não tem os recursos necessários que estão a ser requisitados. O consumo de memória RAM também irá ser uma das limitações quando se trabalha com virtualização, visto que cada máquina irá obrigatoriamente ocupar uma parte (Tulloch, 2010, pp. 6-9) .

Windows Server 2012 R2

O sistema operativo Windows Server 2012 R2 é o resultado de mais de duas décadas de aperfeiçoamentos e melhoramentos em relação a sistemas operativos servidores da Microsoft. Nas suas origens teve o Windows NT, o Windows 2000, o Windows Server 2003 e Windows Server 2003 R2, o Windows Server 2008, o Windows Server 2008 R2 e por fim o Windows Server 2012. Esta cadeia de evolução chegou à era da virtualização no Windows Server 2008 com a implementação do Hyper-V no sistema operativo e atingiu uma grande eficiência e estabilidade até chegar ao Windows Server 2012 R2 bem como novas funcionalidades. (Walat, 2017)

Este sistema operativo tem várias versões, sendo as mais utilizadas as de Standard e Datacenter. A edição de Standard limita o número de instalações em uma máquina física e duas máquinas virtuais e a edição Datacenter não tem limitações de licenças, mas é mais dispendiosa (Rosa, 2013, pp. 15-16). O Windows Server 2012 R2 é um sistema que pode ser configurado como o administrador entender. Define-se *role* como um serviço que o servidor pode conceder, assim quando se faz a sua instalação são instaladas as *roles* necessárias definidas pela Microsoft. No entanto todas as restantes *roles* são instaladas conforme se precisar, desta forma

consegue-se uma melhor gestão da máquina visto que não se irá instalar todas as funcionalidades existentes. Toda a gestão pode ser feita de forma centralizada por uma consola chamada *Server Manager* que irá ter as ferramentas disponíveis ou a possibilidade de as instalar, de monitorizar eventos, ver serviços e aceder de forma local ou remota (*ibidem*, pp. 50-51). É possível visualizar a interface desta consola na figura abaixo.

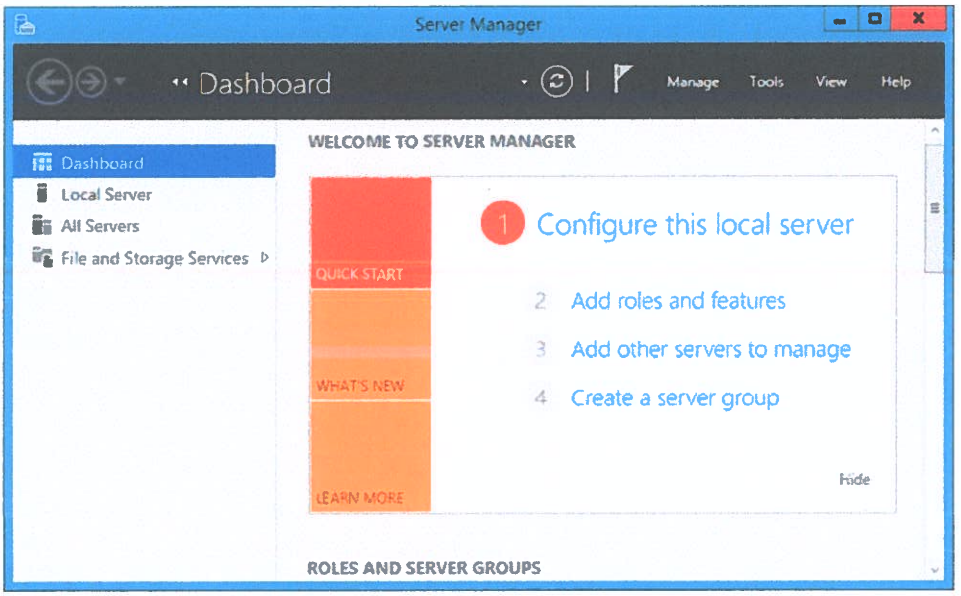

Figura 3 - Server Manager
(Microsoft, 2013)

Uma das razões para se investir na virtualização é a facilidade com que se consegue criar uma infraestrutura e desenvolvê-la, assim o Windows Server 2012 R2 oferece uma escalabilidade, performance e alta disponibilidade que não é comparável até esta versão, assim os melhoramentos feitos em relação ao Hyper-V foram diversos. A nível de processamento é possível ter até 64 CPUs virtuais, uma melhoria de quatro vezes em relação ao Windows Server 2008 R2. A nível de storage tem-se um novo formato para os discos virtuais chamado VHDX, que suporta até 64 TB por disco virtual enquanto o seu antecessor VHD apenas ia até aos 2 TB. Além da vantagem de armazenamento também oferece novos mecanismos para evitar a corrupção de dados em caso de falha de energia. A possibilidade de aumentar ou diminuir o tamanho do disco (VHD ou VHDX) com a MV ligada oferece uma grande vantagem uma vez que não é necessário desligar a máquina (Microsoft, 2013, pp. 8-13).

É possível fazer um *cluster* virtual que contenha vários VHDX, mas apenas seria visto em rede como um único disco com a vantagem de quando houvesse uma falha com um dos discos outro iria assumir as suas funções. Quando se fala em cluster fala-se em alta disponibilidade, o Windows Server 2012 R2 oferece uma opção chamada *Failover Clustering*, que agrega um grupo de servidores de forma a quando um nó do cluster vai abaixo, outro irá tomar as suas funções. Esta funcionalidade admite a transformação de discos em *Cluster Shared Volume* (CSV), permitindo a vários nós acederem ao mesmo disco ao mesmo tempo. Além destes melhoramentos, o sistema operativo oferece a nível de rede uma funcionalidade chamada *Network Load Balancing* que permite gerir os pedidos dos clientes pelos vários servidores juntando-os num *cluster*, com o objetivo de atingir um alto desempenho e tempos de resposta baixos. (*ibidem*, pp. 86-106)

Active Directory

O *Active Directory* (AD) é um serviço de diretório desenvolvido pela Microsoft que faz parte da arquitetura do Windows 2000 e dos seus posteriores, sendo atualizado até chegar ao Windows Server 2012 R2. Para se utilizar basta instalar uma *role* chamada *Active Directory Domain Services* (ADDS). O AD é um serviço que tem toda a informação de forma hierárquica sobre os utilizadores, ficheiros, servidores, impressoras e outros objetos de rede. É um sistema que permite fazer a gestão de forma hierárquica porque está organizado da raiz do domínio para o objeto em questão, entende-se por objeto utilizadores, computadores e impressoras. (Microsoft, 2017,b)

A sua estrutura está divida em duas partes: estrutura lógica e estrutura física. A estrutura lógica do AD é formada principalmente à volta do conceito de domínio. Entende-se por estrutura lógica domínios, árvores de domínios, florestas, Unidades Organizacionais e o *Global Catalog*. Sendo respetivamente uma entrada DNS que identifica um grupo de computadores numa rede externa (internet), um conjunto de domínios de um site, um conjunto de árvores de domínio, um objeto do tipo recipiente que ajuda a gerir objetos e a delegar controlo e um catálogo de todos os objetos que existem na floresta com a sua informação e localização exata. Desta maneira, esta estrutura visa em guardar a informação dos objetos de toda a rede e implementar serviços para que esta informação possa ser usada pelos seus utilizadores, computadores e aplicações. A estrutura física é a responsável por tratar da autenticação bem

como a sincronização de todos os controladores de domínio (DC) através de replicação entre DCs. O DC é a máquina que possuí funcionalidades do ADDS, sendo o responsável por tratar da autenticação e apenas pode pertencer a um domínio, no entanto um domínio pode ter vários DC. Existe um tipo de DC chamado *Read Only Domain Controller* (RODC) que apenas tem capacidades de leitura, desta forma a replicação é assíncrona porque é feita apenas dos restantes DC para o RODC. Assim apenas é capaz de autenticar e não de criar nem alterar objetos (Allen, 2013, pp. 9-14, pp. 229-230)

Além deste tipo existem DC com funções únicas denominados por *Flexible Single Master Operator* (FSMO), sendo que três deles existem em todos os domínios e dois em toda a floresta. Os que existem unicamente em toda a floresta são o *Schema Master*, que permite ao DC alterar o *schema* e por defeito é atribuído ao primeiro DC da floresta. O *Domain Naming Master* que tem a capacidade de alterar o espaço de nomes podendo assim adicionar, remover, mover ou alterar o nome de domínios na floresta. Os restantes três existem singularmente em cada domínio. O *Rid Master* que tem como objetivo garantir que o ID de segurança (SID) único composto pelo SID do domínio em que foi criado e do Relative ID (RID) atribuído pelo *Rid Master*. O *PDC Emulator Master* que é o responsável por manter as *passwords* de todos os utilizadores atualizadas em todos os DC bem como da sincronização do tempo no domínio. Por último o *Infrastructure Master* que irá ter a função de manter as referências de objetos de outros domínios, comparando os seus dados com os do *Global Catalog* quando um objeto é atualizado. (*ibidem*, pp. 14-21)

A outra componente da estrutura física além do DC são os *sites* que são uma coleção de sub-redes do AD ligadas entre si que descrevem a conectividade da rede. São compostos por um conjunto de DC ligados entre si. Assim os *sites* são usados para ajudar na gestão, no fluxo de replicação e no acesso a recursos. (Microsoft, 2017, b)

As vantagens que o AD oferece são variadas visto que a nível de segurança utiliza o protocolo de autenticação Kerberos 5 e permite dar permissões a cada objeto. A utilização de políticas de grupo ajuda a gerir os acessos aos recursos visto que podem ser usadas sobre as Unidades Organizacionais. A replicação entre DC oferece uma tolerância a falhas uma vez que caso um falhe, outro DC irá assumir o seu papel garantindo sempre uma disponibilidade dos recursos e de autenticação. A rapidez com que se consegue montar uma floresta com vários domínios oferece uma plataforma de escalabilidade além de se puder criar novos atributos ou modificar os atributos dos objetos existentes. (Microsoft, 2014)

SQL

Structured Query Language (SQL) é a linguagem de programação padrão universal para bases de dados relacionais. Foi criada em 1974 pela IBM com o nome de SEQUEL (*Structured English Query Language*) sendo uns anos mais tarde revista e passando-se a chamar SQL. Com o crescimento que obteve devido à sua capacidade de acesso e manipulação de dados foram-se criando várias versões desta linguagem, tendo sido considerada a linguagem padrão para arquiteturas relacionais pelo *American National Standard Institute* (ANSI) em 1986 (Beaulieu, 2009, p. 7).

Existem vários Sistemas de Gestão de Base de Dados (SGBD) que são *softwares* responsáveis por gerir uma base de dados através de uma interface, facilitando aos clientes o processo de construção e manipulação dos dados entre aplicações. Existem vários SGBD, sendo o escolhido para este trabalho o Microsoft SQL Server. Desde o SQL Server 2014 que é possível conectar-se com o Microsoft Azure garantindo mecanismos de alta disponibilidade em caso desastre, fazendo o *backup* pela plataforma Azure. (Microsoft, 2017, a)

Entende-se por base de dados relacional um conjunto de tabelas (com linhas e colunas) que está ligada entre si através de identificadores, que no caso do SQL são chaves primárias e estrangeiras. Desta forma, se as ligações forem bem-feitas consegue-se gerir enormes quantidades de dados. As instruções de SQL podem ser divididas em duas partes principais: *Data Definition Language* (DDL) e *Data Manipulation Language* (DML). Os DDL são instruções que afetam a estrutura da base de dados, como criar, remover ou alterar tabelas. Alguns exemplos são os comandos *Create*, utilizado para criar uma base de dados ou uma tabela. O comando *Alter* para alterar a estrutura da tabela e o comando *Drop* para apagar uma tabela ou uma base de dados. Os DML são instruções que afetam os dados existentes na base de dados como adicionar, apagar ou editar campos da tabela. Alguns exemplos são o comando *Select*, usado para recuperar dados da base de dados. O comando *Insert* para inserir dados na base de dados. O comando *Update* para atualizar os dados e o comando *Delete* para apagar os dados da base de dados (Limeback, 2008, pp. 6-19).

ADO

ADO.NET é uma família de tecnologias que permite interagir com dados estruturados de forma desconectada. A *Framework* .NET (plataforma de desenvolvimento de aplicações e sistemas da Microsoft), tem uma biblioteca conhecida por ADO.NET criada pela Microsoft e engloba todas as classes necessárias para a comunicação de uma aplicação (por exemplo WPF) com uma base de dados. Assim o ADO.NET gere os dados internos, resultantes unicamente dos dados em memória da aplicação como também os dados externos, como por exemplo uma base de dados relacional ou mais concretamente um servidor SQL. Para ser utilizado em uma aplicação é necessário acrescentar o *NameSpace System.Data*. É uma arquitetura desconectada, visto que só existe ligação à base de dados o tempo necessário para interagir com ela (Patrick, 2010, pp. 3-4).

A Microsoft lançou a *Entity Framework* que assenta no modelo ADO.NET, sendo um modelo que faz a ligação entre uma aplicação e uma base de dados externa. Esta *framework* consegue simular a estrutura e relações de uma base de dados e criá-lo na forma de classes. Assim para aceder a campos específicos ou inserir novos registos, pode-se faze-lo de uma maneira mais prática e limpa. Desta forma, as tabelas e as suas relações irão ser automaticamente instanciadas como variáveis. Pode ser utilizada tanto em linguagens como C# ou Visual Basic. Existem três camadas principais sendo a primeira o modelo conceptual, A segunda o modelo de armazenamento e a terceira camada de mapeamento. (*ibidem*, pp. 213-218)

Para melhorar o desempenho desta *framework* com a interação dos dados, a Microsoft desenvolveu um modelo chamado LINQ, *Language Integrated Query*. É um modelo que permite fazer *queries* do estilo SQL de forma a interagir com os dados existentes numa base de dados, em objetos ou coleções .NET. Com a utilização do LINQ to *Entities*, as *queries* vão ser feitas para as *Entities* e serão convertidas automaticamente para SQL sem que o utilizador se aperceba, assim esta linguagem vai simplificar a maneira como se interage com os dados. (Tutorials Point, 2015, p. 93)

Cloud Computing

O conceito de *cloud computing* é vasto, porém através do *National Institute of Standards and Technology* (NIST) define-se serviços em nuvem em três tipos: *Software as a Service* (SaaS), *Platform as a Service* (PaaS) e *Infrastructure as a Service* (IaaS). Bem como as suas formas de implementação através de *Private Cloud*, *Community Cloud*, *Public Cloud* e *Hybrid Cloud*. Por fim a definição do NIST esclarece as cinco caraterísticas fundamentais para se considerar uma *cloud*: *on-demand self-service*, *broad network access*, *resource pooling*, rápida elasticidade e serviços medidos (Badger *et al.*, 2011, pp. 1-3).

A *cloud* rege-se por uma arquitetura de vários inquilinos para que cada utilizador esteja isolado dos restantes. Desta maneira, vários utilizadores podem aceder a vários serviços ao mesmo tempo. O *SaaS* é um serviço que permite a utilizador usar aplicações alojadas em uma infraestrutura *cloud*. O fornecedor deste serviço fica encarregue do *hardware* e do *software* necessário para fornecer o serviço. O cliente apenas pode configurar algumas definições da aplicação. Geralmente é acedido através de um *browser*. Tem a vantagem que pode ser acedido através de qualquer lugar e não existe a preocupação com licenças de *software* ou com atualizações, apenas paga conforme a utilização e o tempo. *PaaS* é um serviço destinado para o utilizador que apenas se quer preocupar com o desenvolvimento de aplicações ou para um ambiente de testes, visto que não possui nenhum controlo sobre a infraestrutura (como servidores, sistema operativo, rede). Apenas têm total controlo sobre as configurações da aplicação que desenvolve. Tem como vantagem a rapidez com que obtêm uma plataforma com os *softwares* e ferramentas necessárias para o desenvolvimento de uma aplicação. *IaaS* é um serviço que oferece ao cliente a capacidade de alugar máquinas virtuais. Assim, o cliente tem controlo total sobre os *softwares* como o sistema operativo ou aplicações. Apesar de não ter controlo sobre a infraestrutura física em que assenta a sua estrutura, tem livre arbítrio sobre caraterísticas como o armazenamento (dependendo do fornecedor) e aplicações desenvolvidas. A principal vantagem deste serviço é o custo porque não existe um grande investimento de capital para a obtenção da infraestrutura. A rapidez com que se obtêm uma infraestrutura segura e pronta a usar também é bastante fascinante. Os vários tipos de implementação de *cloud* definem-se por *Public Cloud*, a infraestrutura de *cloud* é presenteada para qualquer pessoa. A

infraestrutura fica no local do fornecedor. Na *Private Cloud* a infraestrutura de *cloud* é apenas para uso exclusivo de uma organização (que pode ter vários utilizadores). A infraestrutura pode existir no local da empresa ou do fornecedor. Na *Community Cloud* a sua utilização é limitada para uma comunidade exclusiva composta por organizações que tem interesses partilhados entre si, por exemplo os Hospitais. O último tipo de implementação é a *Hybrid Cloud* que é composta por dois ou mais tipos de cloud. Por exemplo, a utilização conjunta de *Cloud* Privada e quando necessário, graças a um pico de utilização, a ligação com uma *Cloud* Pública para a utilização de mais recursos. Apenas se pode considerar uma *cloud* caso esta tenha as seguintes caraterísticas. *On-demand self-service*, a capacidade de o utilizador poder em qualquer altura ajustar configurações computacionais como armazenamento em rede de maneira automática sem nenhuma interação humana. *Broad network access*, a capacidade de o utilizador entrar na *cloud* em qualquer lugar com internet através de vários dispositivos como o computador ou o telemóvel. *Resource pooling*, os recursos computacionais que o fornecedor oferece estão agrupados em uma *pool* que serve para vários clientes através da arquitetura referida anteriormente. Por exemplo memória, processamento ou banda larga. Rápida elasticidade, a alta velocidade com que o cliente ajusta os seus recursos permite uma alta escalabilidade. A ultima caraterística para ser considerada uma cloud são os Serviços medidos, tudo o que for utilizado pelo cliente tem de ser pago. Por isto, existem ferramentas que ajudam a monitorizar e controlar de forma otimizada o uso de recursos. (Mell, 2011, pp. 2-3)

Contextualização

O tema enquadra-se na tecnologia atual uma vez que a aplicação desenvolvida faz a gestão de uma biblioteca num ambiente virtualizado, o que permite que apenas com uma máquina física consegue-se ter um sistema responsável por fazer o trabalho de uma sala de servidores. Esta aplicação poderá ser acedida pelos membros do domínio que possuam as permissões para tal.

A virtualização não é novidade, no entanto isso não impede que o seu crescimento tenha vindo a aumentar cada vez. Anualmente, o número de empresas que decide utilizar esta tecnologia nas suas áreas tem vindo a aumentar, visto que os benefícios que esta oferece vão desde a redução de custos imediatos até às despesas de manutenção.

Através da virtualização consegue-se criar infraestruturas capazes de suportarem qualquer tipo de rede ou sistema computacional. Uma vez que ao virtualizar servidores ou Desktops, utiliza-se os recursos todos da máquina física, algo que não acontece nos servidores normais. Assim, a virtualização está cada vez mais presente ou pode-se equacioná-la em várias áreas.

Em suma, pretende-se que sejam elucidativos os benefícios que esta tecnologia demonstra, uma vez que permite a hipótese de se criar infraestruturas e sistemas que apenas seriam exequíveis se houvesse várias máquinas para tal.

Desenvolvimento

Máquinas Virtuais

Este tipo de tecnologia requer um computador que tenha recursos para sustentar as máquinas virtuais, para tal irá assentar num portátil Lenovo com o sistema operativo Windows 10 *Education*, com um processador i5-2540M de 2.60GHz, um disco *SSD* com 250 GB e 16 GB RAM.

O laboratório será constituído por seis máquinas virtuais, sendo responsáveis por:

- DC: Máquina irá ser o Controlador de Domínio, terá as funções do *Active Directory Domain Services*, sendo a responsável por autenticar todas as máquinas no domínio bem como gerir os acessos aos recursos.
- *Router*: Máquina responsável pela divisão da rede externa e interna, sendo que irá distribuir o tráfego pelas restantes máquinas e irá ser a porta de saída para o mesmo.
- Storage: Responsável por guardar os dados dos servidores de SQL. Em cenários empresariais deveria haver duas máquinas de Storage, mas devido à falta de espaço físico apenas haverá uma. Para contornar esta limitação, esta máquina terá dois discos virtuais numa storage pool em modo *Mirror*. De seguida este disco irá ser partilhado para os dois servidores de SQL.
- SQL: Irá alojar o servidor de SQL que terá a base de dados da aplicação. Faz parte de um *Failover Cluster* de forma a conseguir alternar em caso de falha com a máquina SQL2. Para tal é necessário um *cluster* e um *storage* partilhado.
- SQL2: Segunda máquina de servidor SQL, que fará parte do Failover Cluster.
- Cliente: Máquina que irá ter o Visual Studio e a aplicação, responsável por se ligar à base de dados através da aplicação.

O *Hypervisor* utilizado será o VMware Workstation Pro visto que foi o escolhido durante o leccionamento da Pós-Graduação.

Para criar as máquinas virtuais basta abrir o Hypervisor, criar nova máquina virtual, selecionar o ISO a utilizar, selecionar o sistema operativo, o nome e a diretoria da máquina, o tamanho do disco e como o pretendemos gravar (em vários ficheiros ou apenas num), para o laboratório irá guardar-se apenas num ficheiro. O sistema operativo utilizado foi o Windows Server 2012 R2 Datacenter para todas as máquinas exceto a máquina cliente que tem o Windows 8.1 Pro. Na tabela abaixo fica demonstrado as especificações de cada MV:

Tabela 1 - Caraterísticas Máquinas Virtuais
(Fonte do Autor)

Máquinas Virtuais	**Disco**	**RAM**	**Placas de Rede**	**IPv4**
DC	60 GB	2 GB	Host-Only	10.0.1.10/24
Router	60 GB	1 GB	NAT Host-Only	Auto 10.0.1.11/24
Storage	60 GB 250 GB 250 GB 20 GB 20 GB	2 GB	Host-Only Host-Only	10.0.1.13/24 11.0.1.13/24
SQL	75 GB	2 GB	Host-Only Host-Only Host-Only	10.0.1.12/24 11.0.1.12/24 12.0.1.12/24
SQL2	75 GB	2 GB	Host-Only Host-Only Host-Only	10.0.1.14/24 11.0.1.14/24 12.0.1.14/14
Cliente	60 GB	4 GB	Host-Only	10.0.1.15/24

Os discos utilizados são todos com memória dinâmica, por isso é que é possível atribuir valores tão altos.

Na sua instalação e configuração inicial foi utilizada a conta de administrador em todas. Posteriormente, as contas utilizadas para iniciar sessão e configurá-las encontram-se na tabela abaixo.

Tabela 2 - Contas de cada Máquina Virtual
(Fonte do Autor)

Máquinas Virtuais	Login	*Password*
DC	PROJETO\administrator	p@ssw0rd
Router	PROJETO\administrator	p@ssw0rd
Storage	PROJETO\administrator	p@ssw0rd
SQL	PROJETO\clusteradmin	P@ssw0rd
SQL2	PROJETO\clusteradmin	P@ssw0rd
Cliente	PROJETO\alex	P@ssw0rd

Para a instalação do sistema operativo basta apenas escolher o formato do tempo e do teclado para português e escolher a opção com o GUI e selecionar o disco que se pretende instalar o sistema operativo. Foi escolhido a versão Datacenter porque oferece um número ilimitado de licenças.

Controlador de Domínio

Depois de instalado o sistema operativo é necessário fazer algumas configurações antes de se promover a máquina a DC. As seguintes alterações são feitas em todas as máquinas com o respetivo nome e IP.

Primeiro tem de se alterar o nome da máquina, para tal na janela do Server Manager, carrega-se em Local Server e de seguida no nome da máquina. Na nova janela carrega-se em alterar e escreve-se o nome que se pretende atribuir à máquina, no caso DC. A máquina vai reiniciar e quando ligar já terá o nome mudado. O próximo passo será mudar o IP da placa, para tal voltasse à janela do Server Manager, Local Server e na placa que se pretende alterar, no caso do DC só existe a ethernet0 e à sua frente carrega-se em IPv4 *address assigned by* DHCP, IPv6 *enabled*. De seguida, altera-se o endereço IPv4 passar deixar de receber um IP automático e passar a receber um determinado IP que será 10.0.1.10 com uma mascara de rede 255.255.255.0, com o *Default Gateway* de 10.0.1.11 que irá ser o IP da máquina Router e por fim coloca-se o IP de DNS como 127.0.0.1, para indicar que será a própria máquina como servidor de DNS primário. Depois de alterar o IP, convém mudar o nome da placa de rede para algo mais expressivo, para tal na janela de *Network Connections* basta com o rato direito sobre a placa e selecionar a opção *"Rename"* e atribuir o nome de Gestao.

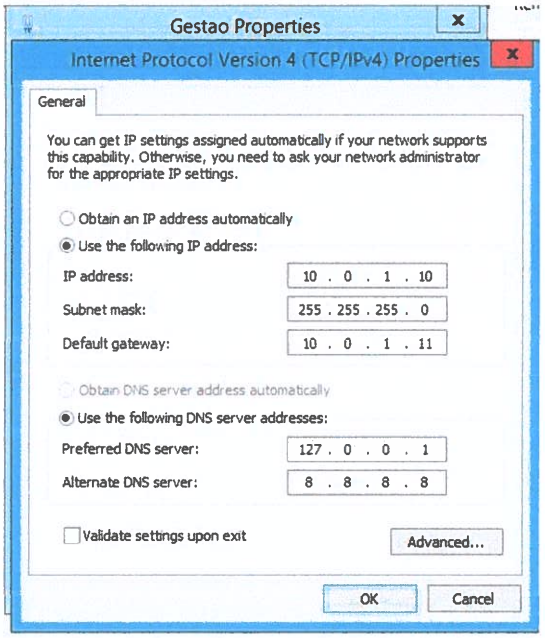

Figura 4- Configuração de rede DC
(Fonte do Autor)

De seguida tem que se ativar os serviços de ICMP na *firewall*, para tal basta pesquisar por *Advanced Security*. Agora seleciona-se a opção *Inbound Rules* e do lado direito, no painel *Actions*, carrega-se em *Filter by Group* e escolhe-se a opção *Filter by File and Print Sharing*. Agora basta selecionar todas as regras que aparecem e com o botão direito do rato carregar em

Enable rule. Esta operação é para permitir às máquinas terem comunicação umas com as outras, por exemplo para conseguirem realizar o comando *ping* umas às outras.

Os *updates* foram feitos antes de se proceder a qualquer tipo de instalação, sendo atualizados posteriormente.

A última configuração que é feita em todas as máquinas é a instalação das *VMware Tools*, para tal na parte superior da janela do *Hypervisor*, em VM e escolher a opção que diz *Install VMware Tools*, que irá permitir uma melhor resolução dentro das máquinas, a criação de pastas partilhadas entre o *Host* e os *guests*, a utilização controlos como *copy paste* do *Host* para a MV entre outras melhorias.

Agora pode-se começar a instalação dos serviços do Active Directory, para tal abre-se a PowerShell e escreve-se o comando seguinte que irá incluir as instruções de PowerShell para a criação de florestas, domínios e promoção de computadores a DC:

Add-WindowsFeature AD-Domain-Services –IncludeManagementTools

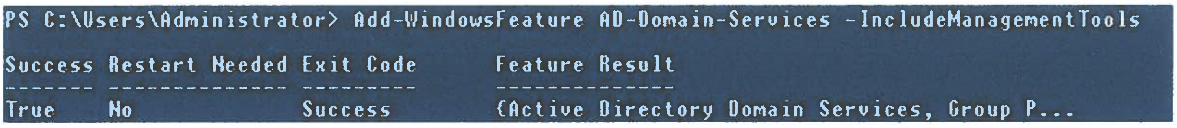

Figura 5 - Instalação ADDS
(Fonte do Autor)

Até esta altura apenas foram instalados os *Active Directory Domain Services* (ADDS), falta ainda promover a máquina a controlador de domínio numa nova floresta, para tal efetua-se o seguinte comando: *Install- ADDSForest – DomainName "projeto.local"*

Vai ser necessário introduzir e confirmar a *password* de recuperação dos serviços do AD caso aconteça alguma falha. Por fim basta inserir a tecla "A" de forma a aceitar todos. É neste passo que irá ser criado e configurado o servidor de DNS.

Figura 6 - Criação de domínio
(Fonte do Autor)

Depois de reiniciar, irá entrar como administrador no domínio projeto.local.

Como a *firewall* do DC não reconheceu o tipo de rede em que estava, uma vez que aponta que está numa rede pública é necessário ir as configurações da placa de rede, IPv4, na parte avançada, no separador DNS e acrescentar o nome do domínio na caixa de texto DNS *Suffix for this connection*. Depois de reiniciar o DC irá aparecer na janela de *Local Server - Domain: On* conforme se poder verificar na Figura 7. Por fim, verificar se todas as regras de entrada de *Filter by File and Print Sharing* estão habilitadas.

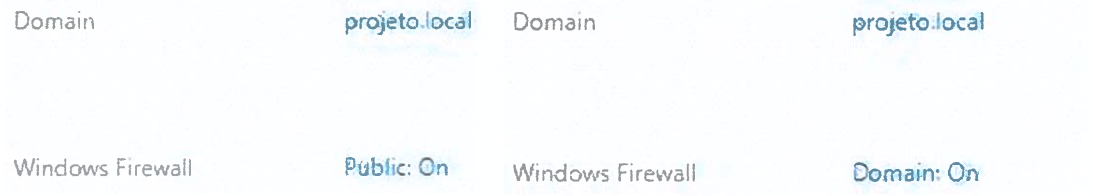

Figura 7 - Public / Domain
(Fonte do Autor)

Router

Como foi referido anteriormente, foram feitas as configurações necessárias antes da instalação de qualquer serviço. Assim, depois da instalação do sistema operativo alterou-se o nome da máquina para Router, e de seguida na placa Host-Only configurou-se de forma a ficar na mesma gama que a outra máquina, com o IP 10.0.1.11, o mesmo IP do *Gateway* do DC, com uma mascara de rede 255.255.255.0 e o *Default Gateway* em branco. Na parte de DNS colocou-se o endereço do controlador de domínio 10.0.1.10. Depois de alterar o IP, alterou-se o nome das placas sendo a que foi configurada chamada de Gestao e a outra de Externa. Pode-se ver estas configurações de rede na Figura VIII.

O Router será a máquina responsável por trazer internet através da placa NAT e irá dar acesso as máquinas virtuais, filtrando o tráfego, visto que apenas irá haver um caminho de entrada e saída da rede do domínio. Assim coloca-se em todas as outras máquinas o IP do Router, menos na própria, na zona de *default gateway*.

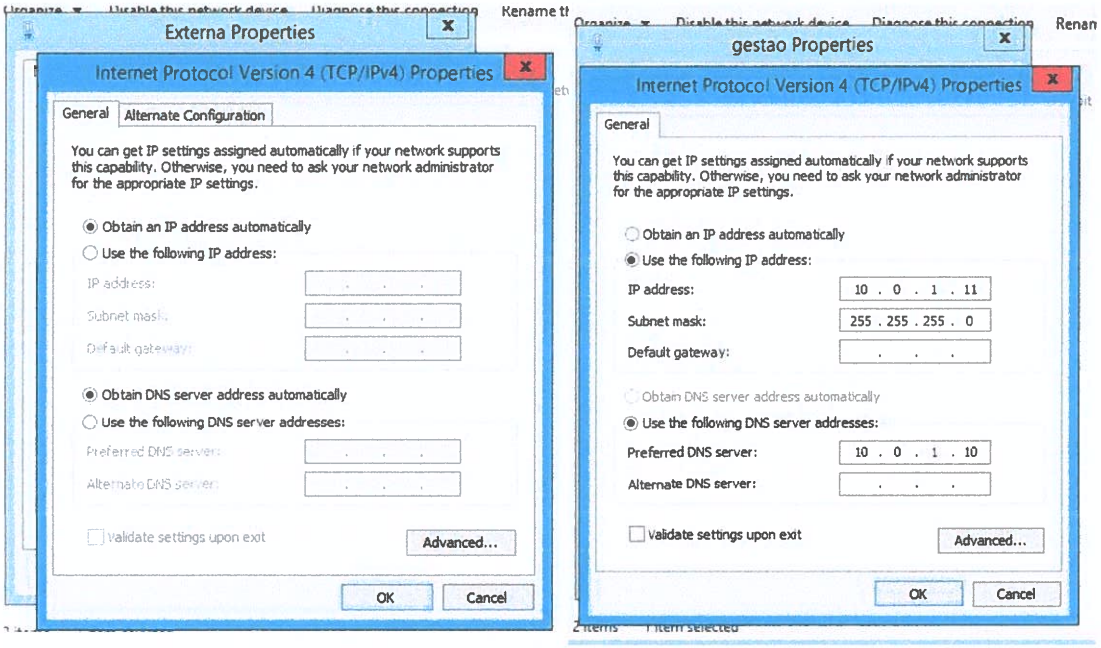

Figura 8 - Configuração de rede Router
(Fonte do Autor)

De seguida habilitar as regras de entrada referidas no subcapítulo anterior de *"Filter by File and Print Sharing* "e instalar as *VMware Tools*. Pode-se verificar as regras habilitadas na Figura IX. De seguida pode-se adicionar a máquina ao domínio, para tal basta carregar janela Local Server e no nome do *workgroup*. Irá abrir uma janela onde se pode escolher e inserir o domínio projeto.local. Será necessário introduzir a password de administrador e de seguida a máquina irá ser reiniciada. Os *updates* também foram feitos nesta fase, tal como nas restantes máquinas.

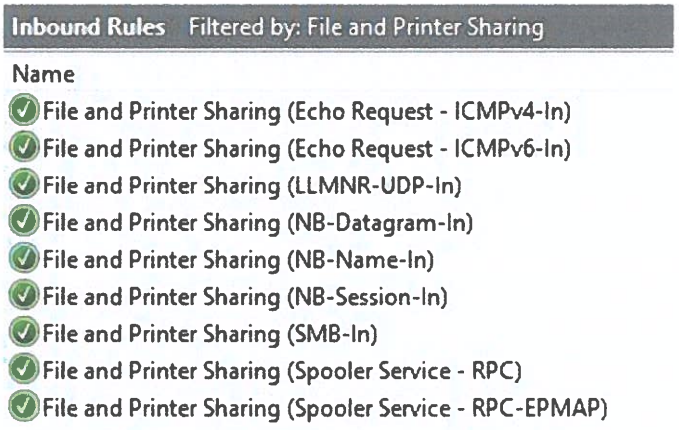

Figura 9 - Regras Firewall
(Fonte do Autor)

Depois de entrar como administrador no domínio convém verificar se o mesmo erro do capítulo anterior não aconteceu. Agora pode-se instalar o serviço de *Remote Acess*, para tal na janela de *Server Manager* escolher "*Add Roles and Features*", selecionar *next* até as *Server Roles*, nesta secção selecionar a *Role* de *Remote Acess* e depois na página *Select Role Services* ativa-se a caixa que diz *Routing* e adicionar as *features* que aparecem. Por fim, carregar *next* e instalar.

Agora falta configurar o serviço, para tal em *Tools*, no *Server Manager* escolher a opção *Routing and Remote Acess*. Na janela que aparece carrega-se com o botão direito em cima do servidor e escolhe-se a opção *Configure and Enable Routing and Remote Acess*. Na janela de configuração escolher a opção *Network Address Translation* (NAT), que irá permitir as restantes máquinas virtuais aceder a internet através de apenas um IP público. De seguida, escolhe-se a placa de rede que irá estar ligada a rede externa, neste caso será a placa que se atribuiu o nome de Externa. Quando terminar a instalação irá ter acesso à internet na máquina DC, visto sendo a única que está no domínio ainda.

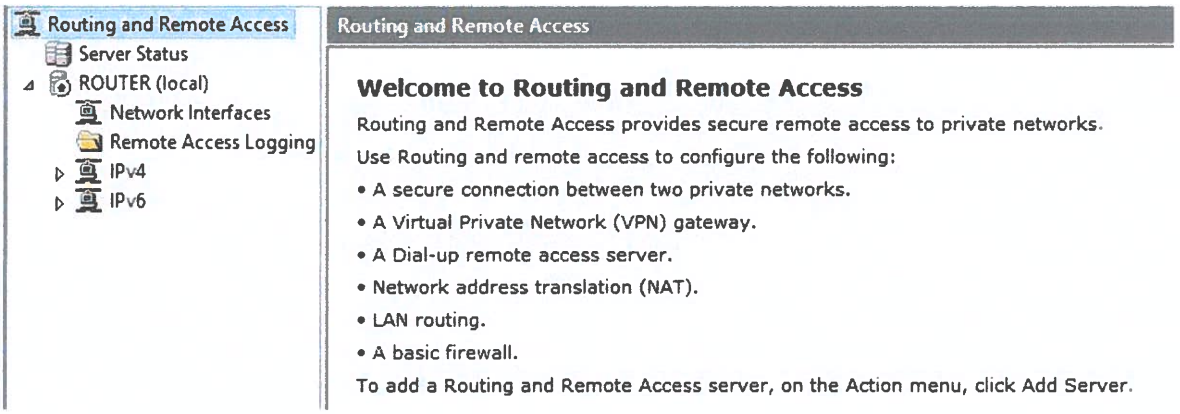

Figura 10 - Routing and Remote Acess
(Fonte do Autor)

Storage

A máquina Storage irá ter cinco discos. Um deles para o sistema operativo, outros dois irão ser adicionados a uma *Storage Pool* onde irão fazer *Mirror* um com o outro com o objetivo de proteger os dados provenientes dos servidores de SQL. O mesmo irá acontecer para os restantes dois discos que irão estar numa *Storage Pool* em modo *Mirror* com o objetivo de

proteger os dados do disco destinado ao *cluster*, o *Quorum Disk*. Assim consegue-se assegurar uma réplica de todos os dados em caso de falha, visto que haverá sempre um disco de *backup*. Com a utilização de Storage Pool, consegue-se agrupar vários discos físicos em uma unidade lógica, criando e mapeando posteriormente um volume que terá por trás os discos existentes na Pool.

Portanto antes de iniciar tem que se adicionar os discos, para tal carrega-se com o botão direito na máquina virtual, propriedades e seleciona-se adicionar *Hard Disk*, escolhe-se os tamanhos que no caso foram, 2 discos de 250 GB e 2 discos de 20 GB. O disco do sistema operativo foi criado anteriormente e tem 60 GB.

Figura 11 - Discos físicos Storage
(Fonte do Autor)

As configurações referidas anteriormente foram feitas nesta máquina sendo o nome da máquina Storage, os serviços de ICMP da *firewall* ativados e as *VMware Tools* instaladas.

Esta máquina irá ter duas placas de rede Host-Only, uma para estar na rede do domínio e outra que irá partilhar os discos através de iSCSI para os servidores de SQL.

Nas placas alterar o nome de cada uma para Gestao e storageNetwork respetivamente. Na placa Gestao, alterar o IPv4 para 10.0.1.13, com a máscara de 255.255.255.0, com o default

gateway de 10.0.1.11 (Máquina Router) e no endereço do servidor de DNS 10.0.1.10 (Máquina DC). Na placa storageNetwork atribuir o IP 11.0.1.13 com a rede 255.255.255.0. Pode-se visualizar estas configurações na figura XII.

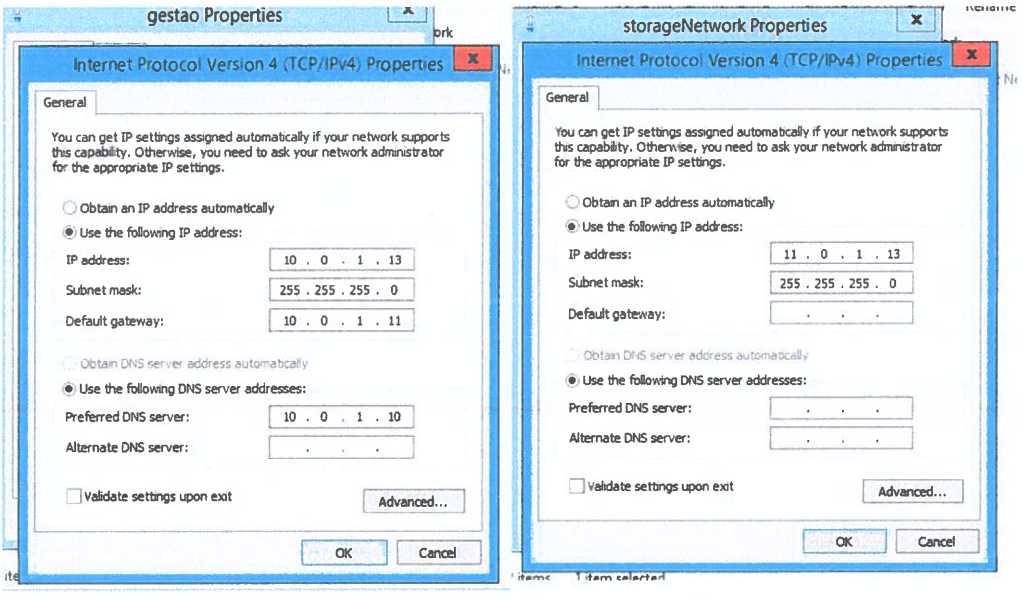

Figura 12 - Configuração de rede Storage
(Fonte do Autor)

Para adicionar ao domínio basta carregar no nome da máquina ou no *workgroup* na janela de *Server Manager* e fazer o mesmo procedimento que a máquina Router.

Para partilhar os discos foi escolhida a opção de *iSCSI*, colocando a máquina como *Target Server*. Para adicionar este serviço na janela de *Server Manager*, *"Add Roles and Features"*, na zona de *Server Roles*, expandir onde diz *"File and Storage Services"* e selecionar *iSCSI Target Server*, incluindo a *feature* que aparece. Depois de a instalação estar concluída é altura de criar as *Pools*, mas primeiro é necessário trazer os discos *online* e inicializa-los usando *Guid Partition Table*, sendo o mais aconselhável para discos em cluster. Para fazer este procedimento, na janela de gestão de discos, botão direito em cima de cada disco e selecionar as opções referidas.

Para criar as Storage Pools, na janela de *Server Manager*, escolhe-se a opção *"File and Storage Services"* e de seguida *Storage Pools*. Apenas existe a opção Primordial, que indica que os discos que não foram utilizados e podem ser adicionados a uma Pool. Para o fazer, botão do lado direito em cima desta opção e escolher *"New Storage Pool"*, dando um nome, escolhendo os discos que se quer adicionar, no caso escolheu-se os discos de 250 GB.

De seguida, tem que se criar um novo disco virtual dentro da Pool, para tal botão direito em cima da recém-criada, e *"New Virtual Disk Wizard"*, escolhendo o nome e se o disco vai ser *Simple, Mirror ou Parity*. O escolhido foi a opção *Mirror*, de forma aos dados serem replicados entre os dois discos físicos, com o tamanho fixo (requisito para os discos em cluster). Agora falta criar o volume, para tal pode-se fazer diretamente depois da criação do disco ou caso se feche a janela, com o botão direito sobre o nome do disco virtual e *"New Volume"*, escolhe-se o tamanho, que irá ser a capacidade máxima do disco, uma Letra de identificação, a formatação NTFS e escolhe-se o nome. Estes procedimentos repetem-se para a criação da segunda Storage Pool até à criação de um novo volume de forma ficarmos com dois volumes chamados DadosClusterMirror e QuorumMirror conforme se pode verificar nas imagens abaixo.

Figura 13 - Storage Pools
(Fonte do Autor)

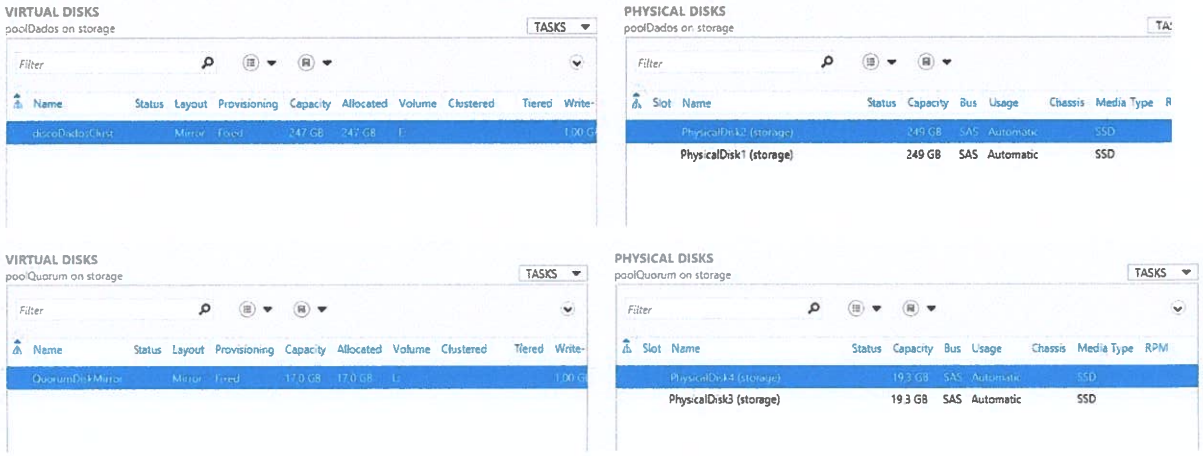

Figura 14 - Discos Virtuais
(Fonte do Autor)

Agora basta partilhar estes volumes por *iSCSI*, para tal voltar a janela de *"File and Storage Services"* e escolher a opção *iSCSI* e carregar na opção para criar um novo disco virtual *iSCSI*. De seguida escolhe-se o disco a partilhar, o nome que este disco irá ter, tamanho, que foi o máximo a expandir dinamicamente, criar um novo *iSCSI Target* atribuindo um nome e na janela *"Add initiator ID"*, escolhe-se os dois servidores para que se quer partilhar os discos,

através do IP da rede storageNetwork. Este processo também se repete para os dois volumes criados de forma a obter o resultado da figura abaixo.

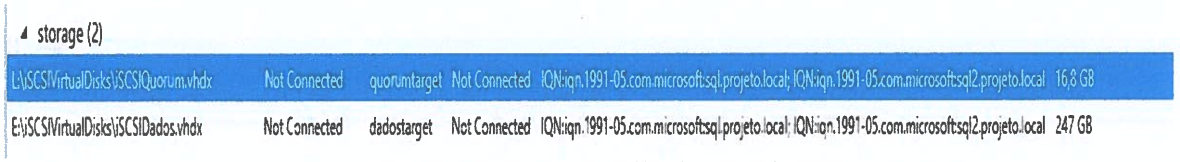

Figura 15 - Discos partilhados por iSCSI
(Fonte do Autor)

SQL

Nesta máquina o nome foi alterado para SQL e as definições ditas anteriormente foram configuradas tais como a abertura de algumas regras de *Firewall* e as *VMware Tools*. Esta máquina terá três placas de rede, uma para o domínio (Gestao) outra para receber os discos iSCSI chamada storageNetwork e outra para as comunicações de *cluster* com o nome clustNetwork que irá fazer *Heartbeat*.

Para a rede de gestão o IP é 10.0.1.12, com uma máscara de 255.255.255.0, com o *Default gateway* de 10.0.1.11 e o endereço de DNS de 10.0.1.10. Para a rede de storageNetwork o IP é 11.0.1.12 e a máscara de rede é 255.255.255.0. Para a rede de clustNetwork o IP é 12.0.1.12 e a máscara 255.255.255.0. Após estarem concluídas as configurações de rede a máquina foi adicionada ao domínio projeto.local. Estas configurações estão demonstradas na figura XVI.

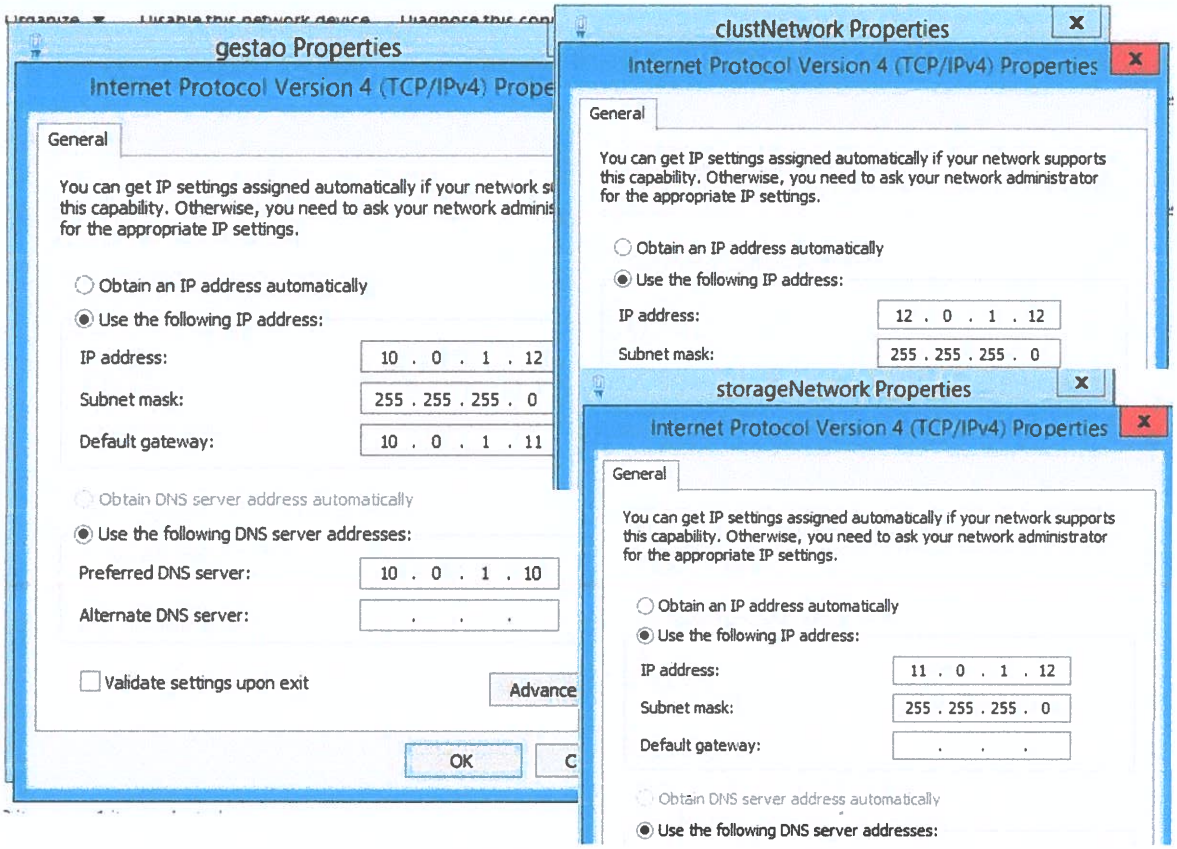

Figura 16 - Configuração de rede SQL
(Fonte do Autor)

Esta máquina e a outra máquina de SQL vão estar em cluster e irão instalar as mesmas *Roles* e fazer as mesmas configurações até a instalação do SQL Server 2016. As *Features* que vão ser instaladas a partir do *Server Manager* são .NET Framework 3.5 e 4.5, *Failover Clustering* e *Multipath I/O* incluindo as *Features* que surjam.

Depois da instalação estar concluída é altura de "receber" os discos partilhados. Para tal, na janela de *Server Manager*, em *Tools*, escolher a opção *iSCSI Initiator*. Irá aparecer uma janela a perguntar se é pretendido ativar o serviço automaticamente sempre que se iniciar o servidor, carrega-se afirmativamente e na zona do target escreve-se o IP da máquina storage, onde irá aparecer os discos partilhados. De seguida no separador *"Volumes and Devices"*, escolher a opção *Auto Configure.* Agora na janela de gestão de discos, irá aparecer os dois volumes, bastando apenas traze-los *online* e inicializá-los. Uma configuração importante é na janela de *Server Manager*, em *"Tools"*, escolher a opção *"MPIO"*, e no separador *"Discover Multi-Paths"* selecionar a opção *"Add support for iSCSI devices"*.

Por volta desta altura convém ir ao DC, e criar o utilizador clusterAdmin, que irá ser o responsável pelas configurações ao nível do *cluster* e do servidor de SQL. Isto implica que tem de ter um nível de permissões ao nível dos administradores, quer a nível local de cada uma das máquinas em questão como também a capacidade de criar objetos no Active Directory, uma vez que a criação do *cluster* e do cluster de SQL irão criar computadores ao nível do AD. Para criar um novo utilizador, no DC, na janela de *Server Manager*, *Tools*, *Active Directory Users and Computers*, selecionar a pasta *Users* e botão do lado direito, criar novo utilizador. Depois de criado este utilizador, com o botão do lado direito sobre ele, propriedades e no separador *"Member of"*, adicionar o grupo *Administrators*. De seguida, no separador *"View"* escolher *"Advanced Features"* e na pasta que diz *"Computers"*, com o botão do lado direito, propriedades, "*Security*" e adicionar o utilizador que foi criado, dando-lhe permissões para criar objetos. Pode-se ver estas alterações na figura XVII.

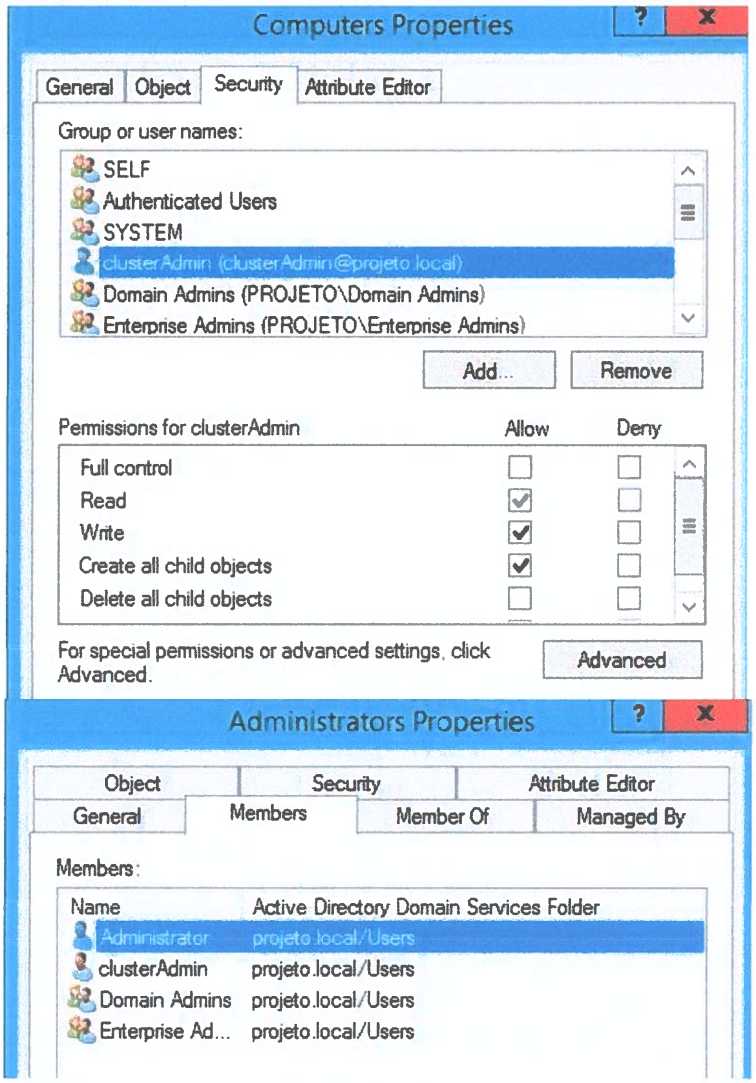

Figura 17 - Permissões a nível de domínio
(Fonte do Autor)

Voltando à máquina SQL, em *"Computer Management"*, expandir a opção *"Local Users and Groups"*, selecionar *"Groups"*, e adicionar o utilizador clusterAdmin ao grupo de *Administrators* local. Este processo tem de se repetir na outra máquina de SQL. Ver figura XVIII.

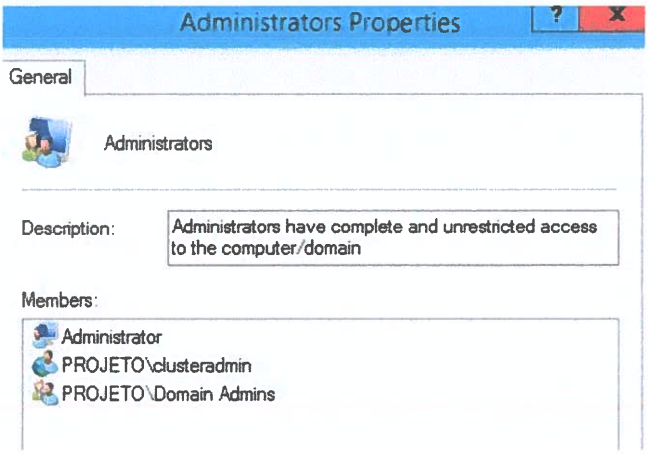

Figura 18 - Permissões ao nível local SQL e SQL2
(Fonte do Autor)

Além das regras que foram referidas em capítulos anteriores neste servidor e no seu parceiro é necessário criar duas regras para permitir ligações remotas à futura base de dados, para tal na janela *"Windows Firewall with Advanced Security"*, criar nova regra que se destine ao TCP com as portas especificadas de 1433 e 1434, aplicando-se apenas ao domínio. De seguida é necessário criar outra regra de entrada, mas desta vez para UDP com a porta 1434, novamente apenas aplicando-se ao domínio.

Inbound Rules

Name	Group	Profile	Enabled	Action	Override	Program	Local Address	Remote Address	Protocol	Local Port
portasSQLTCP		Domain	Yes	Allow	No	Any	Any	Any	TCP	1433, 1434
UDPSQL		Domain	Yes	Allow	No	Any	Any	Any	UDP	1434

Figura 19 - Regras para acesso remoto SQL
(Fonte do Autor)

SQL2

Na segunda máquina de SQL, atribuiu-se o nome de SQL2, sendo as configurações ditas no capítulo anterior repetidas, com o objetivo de ficarem com as mesmas definições.

As três placas configuradas possuem os seguintes IPs, a placa de Gestao com o endereço 10.0.1.14, com uma máscara de rede de 255.255.255.0 e o *Default Gateway* de 10.0.1.11, o endereço de DNS será o mesmo, 10.0.1.10. A placa de storageNetwork tem o IP 11.0.1.14 e uma máscara de 255.255.255.0. A placa clustNetwork tem o IP 12.0.1.14 e a mesma máscara de rede, 255.255.255.0.

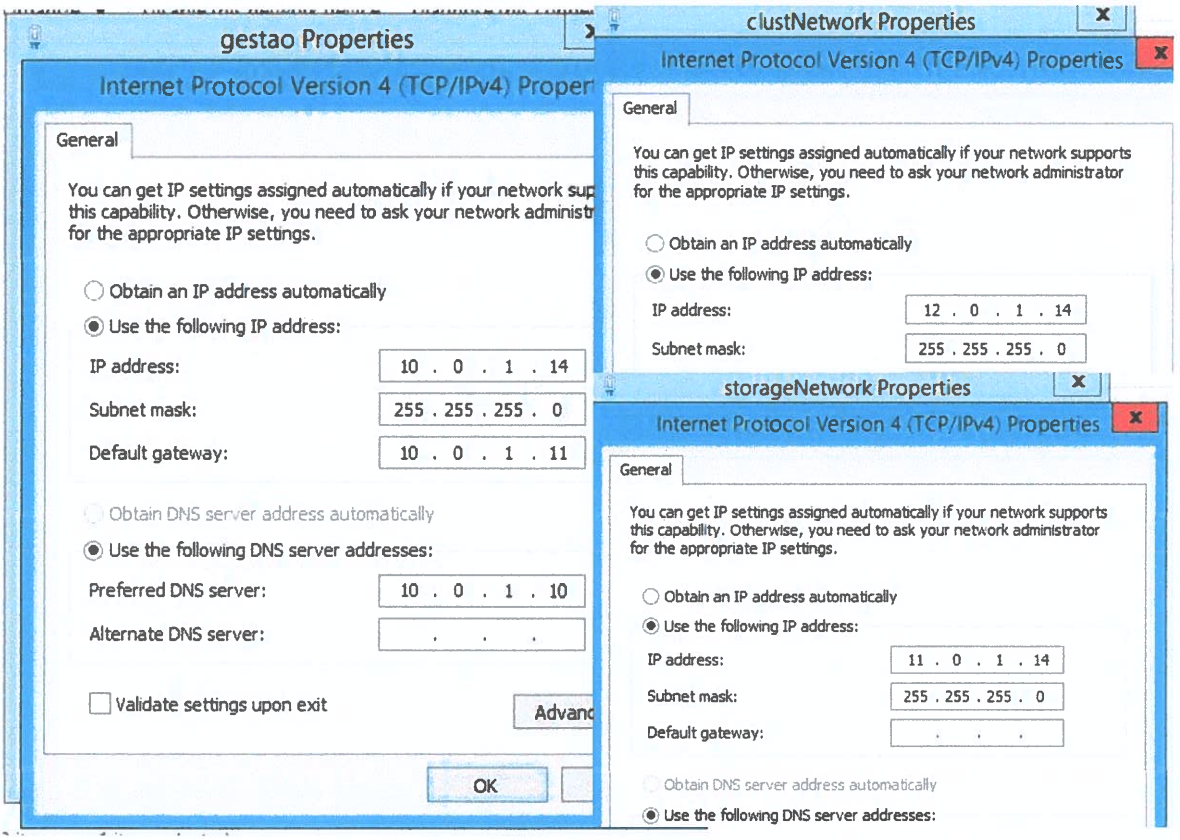

Figura 20 - Configuração de rede SQL2
(Fonte do Autor)

De seguida a máquina foi adicionada ao domínio e posteriormente instaladas as *Features* que foram instaladas no capítulo anterior, não esquecendo de ativar a receção dos discos de iSCSI e de adicionar o utilizador clusterAdmin ao grupo de administradores locais da máquina. Esta máquina terá de estar com os mesmos *updates* do primeiro servidor de SQL. É necessário adicionar as duas regras referidas no capítulo anterior.

Cliente

A máquina cliente seguiu o padrão das restantes máquinas e foram feitas as configurações padrão referidas nos capítulos anteriores, tais como, a mudança de nome, neste caso para Cliente, a ativação de algumas regras na Firewall e a instalação das *Vmware Tools* bem como o Windows Update. Nesta máquina instalou-se o sistema operativo 8.1 Pro x64.

Na placa de rede Gestao, atribuiu-se o IP 10.0.1.15, com a máscara de rede 255.255.255.0, o Default Gateway de 10.0.1.11 e o endereço DNS 10.0.1.10.

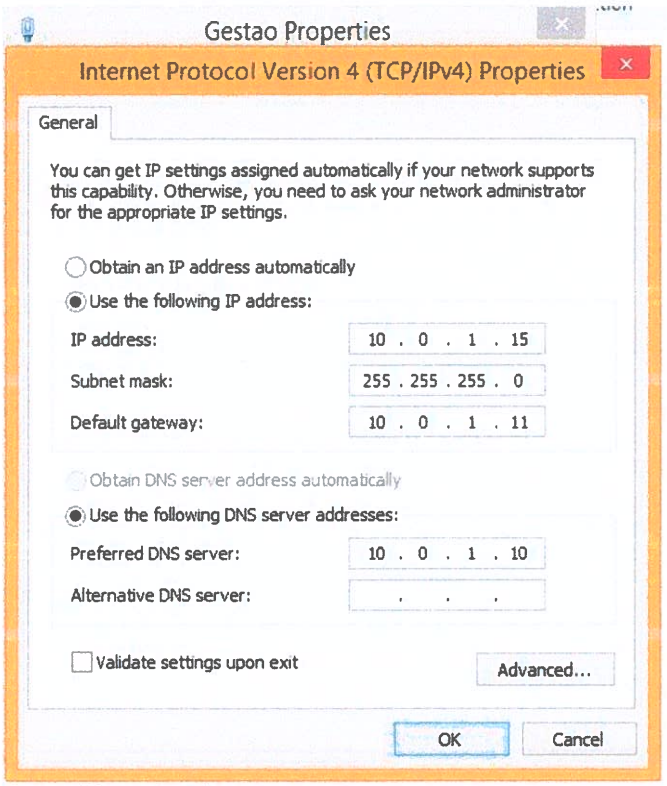

Figura 21 - Configuração de rede Cliente
(Fonte do Autor)

De seguida procedeu-se à colocação da máquina no domínio, para tal com o botão direito em cima de *"This PC"* e escolhe-se a opção propriedades, carregando na janela que se abre em *"Change Settings"*, selecionando o botão alterar e depois escrever o domínio projeto.local na zona apropriada.

Depois de reiniciar e do Windows Update estar concluído, instalou-se o Visual Studio 2017 Community. Será nesta máquina que será desenvolvida a aplicação.

Criação e configuração Cluster

Nesta altura é obrigatório ter as configurações anteriormente referidas feitas nas duas máquinas porque vai-se efetuar a validação do *cluster* e ambas as máquinas têm de estar em pé de igualdade. Na máquina SQL, em *"Tools"*, na janela do *Server Manager*, carregar onde diz *"Failover Cluster Manager"* e na aba do lado direito escolher *"Validade Configuration"*. Aqui irá selecionar-se as máquinas que irão fazer parte do cluster e irá fazer-se todos os testes de validação. Nesta fase é crucial ter tudo configurado, a nível de rede, de permissões além de também ter todos os *updates* em dia, em ambas as máquinas. Depois de terminar, pode-se escolher a opção *"Create Cluster"*, inserindo outra vez as máquinas que irão fazer parte e de seguida vai se atribuir um nome ao *cluster* e um IP que será da gama da rede de gestão, mais concretamente 10.0.1.25 com o nome WindowsCluster.

Depois da criação do *cluster* pode-se confirmar que o *cluster* está funcional visto que caso se carregue em "Nodes" verificamos que estão lá as duas máquinas e ao carregarmos com o botão direito em cima da máquina que tem os discos do seu lado e pararmos o serviço de cluster, indo à zona dos *"Disks"* em *"Storage"*, vemos os discos a passarem de uma máquina para a outra. O *cluster* automaticamente deteta qual é o disco que fica encarregue das suas configurações, neste caso foi o disco mais pequeno. Pode-se verificar qual é visto que fica designado como *"Disk Witness in Quorum"*. Convém fazer algumas alterações nos nomes para tornar mais notórias as configurações do *cluster*. Para tal, em *"Storage"*, *"Disks"*, alterar o nome dos discos para DadosSQL e QuorumDisk respetivamente. De seguida em "Networks", o *cluster* consegue detetar os tipos de rede existentes, visto que na rede que não tem nenhum uso pelo cluster, a rede storageNetwork, aparece a indicação que não esta a ser utilizada pelo *cluster*. Na rede de Gestao permite a comunicação de *cluster* e de clientes e na rede

clustNetwork apenas permite tráfego referente ao *cluster*. Seguindo estas configurações, altera-se os nomes para iSCSI, LAN e HeartBeat respetivamente.

Para terminar as configurações do cluster é necessário tornar o disco DadosSQL num Cluster Shared Volume, desta forma o volume irá desaparecer da zona *"Devices and Drives"* em *"This PC"* e irá ser criada uma pasta dentro do disco C: destinada ao cluster, que irá ter o nome de Volume1. Esta pasta poderá ser acedida ao mesmo tempo pelas duas máquinas, desta forma em caso de falha, mais rapidamente se recupera porque não é necessário desmontar e montar o volume, uma vez que ambas as máquinas têm acesso ao disco, apenas acontece a mudança do Owner Node. Estas configurações podem ser vistas na figura seguinte.

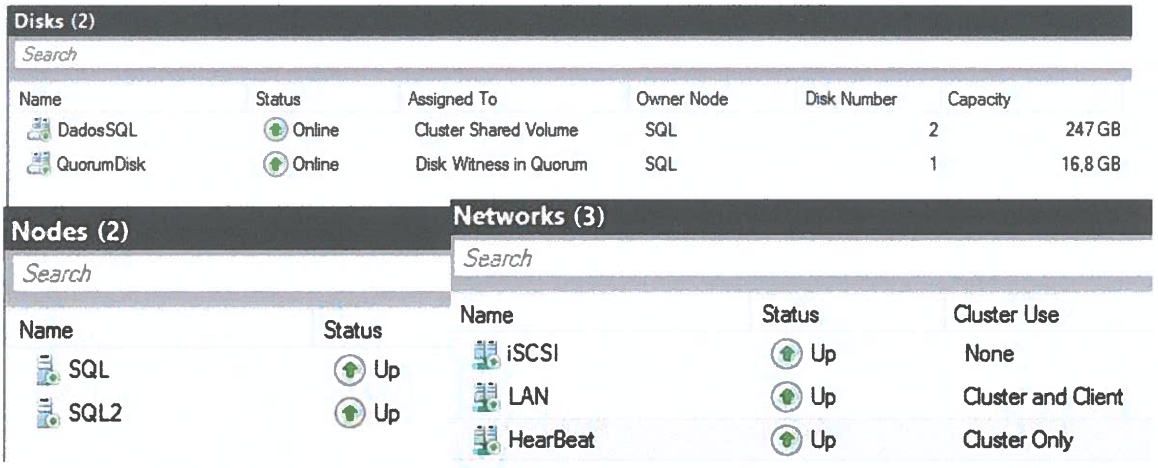

Figura 22 - Configurações Cluster
(Fonte do Autor)

Instalação e configuração SQL 2016

O objetivo do *cluster* é permitir que quando a máquina SQL for abaixo, a máquina SQL2 irá assumir as suas funções automaticamente. Para tal, é necessário instalar o SQL Server 2016 Standard na máquina SQL e na máquina SQL2 adicioná-la a um SQL Server *Failover cluster*.

Na máquina SQL, coloca-se o *ISO* do ficheiro SQL Server 2016 Standard na drive de CD, e escolhe-se a opção *"New SQL Server stand-alone installation or add features to an existing installation"*, insere-se a chave, verifica-se se aparece algum aviso por falta de *updates* ou algum aviso em relação à Firewall. Na janela *"Feature Selection"* escolhe-se as funcionalidades que se quer instalar no servidor, no caso selecionou-se dentro dos *"Database*

Engine Services" as opções *"SQL Server Replication", "Full-Text and Semantic Extractions for Search", "Data Quality Services"* e fora deste grupo as *"Client Tools Connectivity"* e *"SQL Client Connectivity SDK"*, nas diretorias locais da máquina. Na próxima janela é altura de escolher o nome da instância e o nome da rede. A instância é a responsável por tratar de todos os pedidos vindos das aplicações que queiram trabalhar com os dados, incluindo a autenticação. Entende-se por instância um serviço que permite executar vários serviços SQL Server de forma separada, visto que se pode criar múltiplas instancias cada uma com os seus logins e bases de dados estando todas no mesmo servidor. O nome escolhido para a instância foi de SQLCLUSTER e o nome da rede de SQLCLUSTERNET. A seguir irá aparecer para selecionar o disco partilhado, como o QuorumDisk está a ser utilizado pelo *cluster*, apenas se pode escolher o disco DadosSQL. Na janela seguinte atribui-se o endereço IP do *cluster* SQL que será 10.0.1.30. Para se conectar ao servidor de SQL irá ser necessário colocar o IP (ou o nome da rede do cluster) seguido de uma barra invertida e o nome da instância (SQLCLUSTERNET\SQLCLUSTER). De seguida insere-se o *Account Name* e *Password* do utilizador clusterAdmin nos serviços *"SQL Server Agent"* e *"SQL Server Database Engine"*. Na janela *"Database Engine Configuration"*, é altura de escolher o tipo de autenticação para aceder ao servidor SQL. Escolhe-se *Windows Authentication mode* e adiciona-se o utilizador corrente, que é o clusterAdmin. Na mesma janela mas no separador *"Data Directories"* pode-se confirmar que por defeito já está escolhida a pasta que se encontra no CSV.

Depois de instalado o servidor SQL 2016 na máquina SQL, é necessário fazer o mesmo na máquina SQL2. Para tal, depois de ter o *ISO* adicionado, carregar no ficheiro de instalação e escolher a opção *"Add a Failover Cluster Node"*. É obrigatório a máquina em questão ser membro de um *Windows failover cluster*. Para adicionar este nó a um *Failover Cluster SQL* na primeira janela insere-se a chave, e verifica-se que na janela *"Cluster Node configuration"*, a instância que é possível selecionar foi a previamente criada na máquina SQL. De seguida é altura de confirmar que o endereço IP que aparece também foi o mesmo que foi atribuído. Na janela *"Service Accounts"* é necessário inserir a *password* da conta especificada na máquina SQL do utilizador clusterAdmin. As funcionalidades deste nó dependem das funcionalidades e configurações escolhidas inicialmente na outra máquina, visto que apenas se está a adicionar um nó a uma instância já existente.

No final da instalação é possível verificar na janela "*Failover Cluster Manager*", na área dos *"Roles"*, que aparece um role com o nome da instância escolhida e caso se desligue o

nó responsável por este role, ele irá passar para o outro nó. Pode-se verificar esta mudança na figura XXIII.

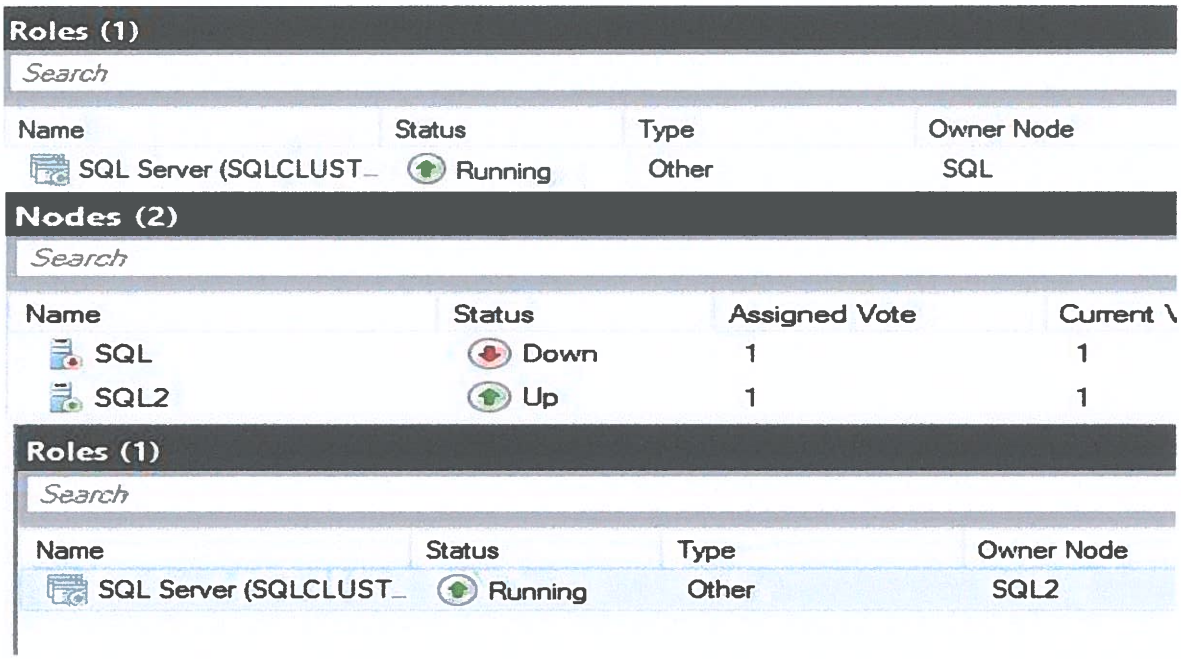

Figura 23 - Mudança de *Owner* em caso de falha
(Fonte do Autor)

Pode-se confirmar no DC, na consola do *"Active Directory Users and Computers"*, em *"Computers"*, que foi criado uma nova entrada para o SQLCLUSTERNET, para tal é que foi necessário dar permissões para a criação de registos no AD ao utilizador clusterAdmin.

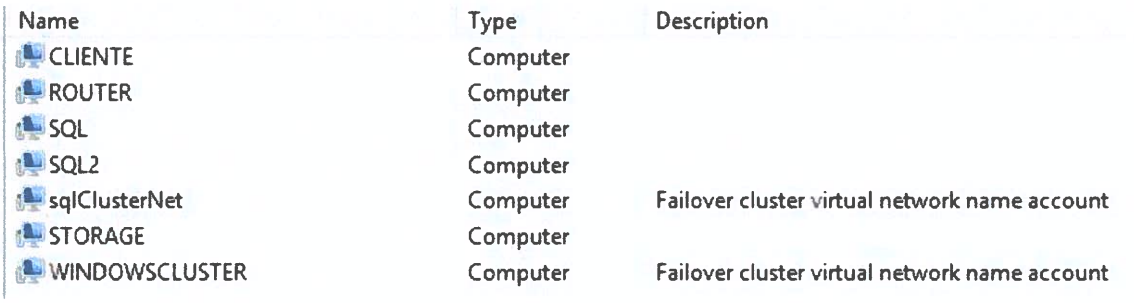

Figura 24 - Criação de Objetos no AD
(Fonte do Autor)

Para concluir as instalações de *software* é necessário instalar o SQL Management Studio, que é um Sistemas de Gestão de Base de Dados que oferece um interface prático e simples para a manipulação de bases de dados. A versão instalada nas duas máquinas foi a 17.

Para os utilizadores conseguirem aceder aos dados da aplicação é preciso dar permissões para o fazerem. Por isso é necessário criar um grupo no AD chamado "Funcionários", que irá ter os membros que poderão aceder à aplicação. Para tal, na máquina DC, voltar a janela *"Active Directory Users and Computers"*, "Users" e com o botão do lado direito escolher novo grupo. Repetir o processo e criar o grupo "adminAplicacao", este será o grupo de administradores da aplicação. Foi criado também um utilizador chamado alex e outro chamado Lima sendo adicionados respetivamente ao grupo adminAplicacao e ao grupo Funcionarios.

Para entrar no SQL Management Studio, basta fazer o Login com o *Server Name* SQLCLUSTERNET\SQLCLUSTER, iniciando a sessão com a conta clusterAdmin. Depois de entrar é necessário criar a base de dados, para tal carregar com o botão do lado direito na pasta *"Databases"* e *"New Database"*. Atribuiu-se o nome biblioProjetoMVVM com o utilizador clusterAdmin como Owner.

Mas, para que os membros dos grupos criados consigam aceder à base de dados é necessário criar um Login. Mas em vez de se criar um Login para cada utilizador, vai ser criado um Login para cada Grupo, desta forma todos os membros do grupo irão ter acesso à base de dados através do grupo em que estão. Para isso, expandir a pasta *"Security"*, *"Logins"* e com o botão do lado direito carregar em "New Login". De seguida, em *Login Name* selecionar que queremos procurar em todo o domínio e que também incluímos grupos na procura e escrever o nome do grupo Funcionarios. Para concluir, do lado esquerdo da janela carregar em User Mapping e escolher a base de dados que queremos associar a este Login que é a biblioProjetoMVVM com o *default schema* de dbo. O que acontece é que será criado um User dentro desta base de dados com o mesmo nome do Login. Para concluir, com o botão do lado direito na base de dados recém-criada, e escolher *"Properties"*. Depois em *"Permissions"*, selecionar o utilizador Funcionarios e atribuir-lhe permissões para se conectar, criar tabelas, apagar, executar, inserir, selecionar e capacidade para atualizar. Assim, todos os membros do grupo Funcionários não terão problema nenhum em ligar-se e executar funções através da aplicação. Repete-se este processo para criar um Login para o grupo adminAplicacao.

Base de Dados

A base de dados a ser utilizada pela aplicação é composta por três tabelas. Os comandos para a criação das tabelas encontram-se no Anexo 1 - Criação das tabelas da base de dados.

A tabela registos é responsável por armazenar todos os registos que acontecem. Entende-se por registo o aluguer por parte de um cliente sobre um livro. É composta por nove campos, tendo uma chave Primária chamada idRegisto que vai incrementando 1 a 1 e duas chaves estrangeiras, uma chamada idLivro que refere a tabela livros e outra chamada idCliente que refere a tabela clientes. O campo dataInicio guarda a data em que é feito o aluguer do livro e o campo dataFim é um campo calculado através do campo dataInicio ambos do tipo Date e indica a data máxima para a entrega do livro, sendo um mês depois da dataInicio. O campo entregar é do tipo *bit* e aceita valores True ou False, sendo utilizado para fazer a gestão de quando o livro está alugado ou por alugar. O campo username indica quem foi o funcionário que efetuou a transação de aluguer, o campo usernameEntrega indica o nome do funcionário que realizou a receção do livro e o campo dataEntrega indica quando foi entregue o livro.

Figura 25 - Tabela registos
(Fonte do Autor)

A tabela livros é onde estão armazenados todos os livros existentes. É composta por uma chave primária chamada idLivro e por vários campos que compõem cada livro. Assim é constituído por um campo nomeLivro, nomeAutor e editora, todos do tipo *varchar*. O campo

foto armazena a capa do livro (imagem) sendo um *varbinary*, o campo estado é do tipo *bit* e vai ser utilizado para saber quando o livro está disponível ou está alugado.

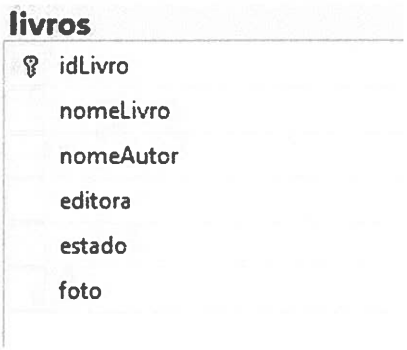

Figura 26 - Tabela livros
(Fonte do Autor)

A tabela clientes é onde estão guardados os nomes de cada cliente bem como o seu número de identificação fiscal. É composta por uma chave primária com o nome idCliente. Também tem um campo *int* que vai permitir fazer a gestão de quantos livros um cliente têm, sendo o máximo de três.

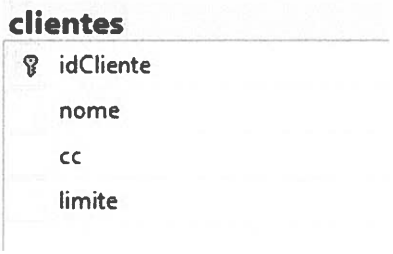

Figura 27 - Tabela clientes
(Fonte do Autor)

Na figura abaixo podemos verificar a relação existente entre as tabelas na forma de um diagrama.

Figura 28 - Relações entre tabelas
(Fonte do Autor)

Aplicação

A aplicação foi desenvolvida na máquina cliente, através do Visual Studio 2017 Community em WPF – *Windows Presentation Foundation* – em C#. A aplicação consiste na gestão que uma biblioteca tem de ter com os seus livros e os respetivos clientes. Assim, os membros do domínio que tenham permissões para entrar na aplicação irão faze-lo através de um Login com Windows Authentication e do grupo do AD em que se encontram. Caso tenham permissões para entrar, será possível criar novos clientes, associar livros aos clientes, fazer a entrega dos livros de cada cliente, atualizar registos, adicionar livros e caso sejam administradores poderão também remover livros ou clientes. A aplicação foi pensada de forma a poder ser implementada, desta forma quando um livro é alugado por um cliente este deixa de estar disponível para qualquer outro e quando o livro for entregue volta a estar livre. Cada cliente apenas poderá ter três livros alugados ao mesmo tempo e cada transação (entrega/aluguer) fica guardada na tabela registos, que poderá ser exportada para PDF. Cada ação fica associada o *username* (da conta Windows do funcionário), de forma a manter-se um registo do que cada funcionário fez.

Durante a criação da aplicação seguiu-se o modelo de MVC, *Model View Controller*, por isso a aplicação encontra-se dividida em três partes. Num *Model* (responsável por tudo o que for interação com os dados e as suas validações), numa *View* (parte que interage com o utilizador, apenas demonstra informação, será o *xaml*) e o *Controller*, responsável por fazer a

ligação entre o *Model* e a *View*, verificando qual método que deve ser utilizado quando se carrega num botão da *View*. No caso do projeto, o *Model* será a classe "ViewModelCliente.cs" e "ViewModelLivross.cs". Cada janela da aplicação será uma *View* e o *Controller* será a classe comandos.cs. Para cada View tem de se especificar que o seu *DataContext* é a "ViewModelCliente" ou a "ViewModelLivros" tanto no xaml como no Code-Behind.

Em todas as Views, no seu Code-Behind, em vez de haver eventos que acontecem quando se carregam em botões, os métodos serão chamados à ViewModelCliente ou à ViewModelLivros, conforme o DataContext da View. Por isso, para qualquer botão na View tem de se fazer a sua declaração no Code-Behind. Esta lógica aplica-se a todos os botões da aplicação.

As páginas seguintes irão explicar excertos de código que compõem a aplicação, lembrando que o código completo se encontra na secção de Anexos.

O login feito na aplicação utiliza os dados da conta Windows dos funcionários, tais como o *username*, *password*, domínio e o grupo em que se encontram, o código para tal encontra-se em Anexos 2 – ViewModelCliente.cs. Se autenticado, o funcionário irá para uma janela onde poderá escolher se vai para a zona de gestão dos clientes ou se vai para o catálogo. É na janela de gestão de clientes que se encontra o núcleo da aplicação visto que é lá que se cria, edita e apaga clientes como também se associa os livros a cada um dos clientes. Para criar um cliente é obrigatório associá-lo a um livro.

Criar Cliente

Desta forma, quando o funcionário carrega no botão para criar o cliente, a classe "comandos.cs" vai indicar o método que o botão pressionado da *View* pretende executar. Este método encontra-se na ViewModelClientes. Por isso, no Code-Behind da *View* "criarCli.xaml" é necessário associar o seu *DataContext* e XAML à ViewModelClientes. Depois precisa-se de fazer o *Binding* do botão. Para tal declara-se a *ViewModelCliente* numa variável global e dentro do construtor da *View* adiciona-se o código abaixo apresentado. Passa-se para o método *insertPessoa* parâmetros como a caixa de texto do nome do cliente, do NIF, o botão *toggle* e a *comboBox*.

```
public ViewModelCliente vmclientes { get; set; }

public criarCli()
{

InitializeComponent();
//associa o xaml e o DataContext à vmclientes
vmclientes = (ViewModelCliente)Resources["nomes"];
this.DataContext = vmclientes;
//Evento que é disparado quando se fecha a janela
vmclientes.ClosingRequest += (sender, e) => this.Close();
//Binding do comando e do método com o botão (xaml)
CommandBindings.Add(new CommandBinding(
comandos.insertCliente,
(sender, e) => vmclientes.insertPessoa(txtCli, txtNif,  toggSwitc, combLivros),
(sender, e) => e.CanExecute = true ));
```

Em comandos.cs encontram-se todos os comandos declarados utilizados por cada botão, pode-se verificar cada um deles em Anexo 4- comandos.cs. Assim, para se inserir um cliente é necessário declarar o comando insertCliente, para tal:

```
public static RoutedUICommand insertCliente = new
RoutedUICommand("insertCliente", "insertCliente", typeof(comandos));
```

É obrigatório no XAML de todas as Views indicar o nome do Controller em questão e associar o comando ao botão da *view*. Assim, em "criarCli.xaml" no XAML, através da propriedade "*Command*":

```
  <Window.Resources>
        <local:ViewModelCliente x:Key="nomes"></local:ViewModelCliente>
</Window.Resources>

<Button    Style="{StaticResource    AccentedSquareButtonStyle}"    Foreground="Black"
BorderThickness="1"  FontWeight="Bold"  Width="150"  Name="btnAtualizar"    Height="40"
Content="Atualizar" Command="local:comandos.insertCliente" Margin="0,0,30,0" ></Button>
```

A *ViewModelCliente* herda do interface *INotifyPropertyChanged*, de forma a puder ser notificada quando existem alterações. Na *ViewModelCliente* é declarado o método. Este método recebe os parâmetros passados, verificando se os campos estão bem inseridos (se as caixas de texto não são *Null*, se o campo NIF apenas é constituído por números, se a *comboBox* tem algum valor escolhido e se o botão toggle está ativo). De seguida, são criadas duas instâncias, do tipo cliente e do tipo registo. Na tabela Clientes, vai-se inserir um novo membro constituído por um idCliente (automático), nome, NIF e limite. A variável limite vai indicar quantos livros o cliente possuí no ativo. Depois do cliente ser inserido, vai-se associar o

respetivo idCliente a uma nova alínea na tabela Registos, de forma a fazer a requisição do livro. Na tabela registos vai ser inserido o idCliente (do cliente criado), o idRegisto (automático), a data inicial (de aluguer), a data máxima de entrega (campo calculado automaticamente através do campo *dataInicio*), o *username* do funcionário que fez a transação e a variável responsável por controlar se o livro que o cliente têm já foi entregue ou não chamada entregar. Caso esteja com o cliente vai estar a *True*, se não vai estar *False*. Por fim é necessário alterar a disponibilidade do livro escolhido, para tal, através do campo idLivro, consegue-se obter o livro escolhido e alterar a variável estado da tabela Livros para True.

```
      public void insertPessoa(TextBox txtCli, TextBox txtNif, ToggleSwitch toggSwitc,
ComboBox combLivros)
        {
            try
            {
                if              (!(string.IsNullOrEmpty(txtCli.Text)              ||
string.IsNullOrEmpty(txtNif.Text))  && toggSwitc.IsChecked == true)
                {
                    int nif;
                    if (int.TryParse(txtNif.Text, out nif))
                    {
                        if (combLivros.SelectedValue != null)
                        {
                            cliente cli = new cliente();
                            registo reg = new registo();
                            cli.nome = txtCli.Text;
                            cli.cc = txtNif.Text;
                            cli.limite = 1;
                            clienteModel.clientes.Add(cli);
                            clienteModel.SaveChanges();
                            ListaClientes.Add(cli);
        //Depois do cliente estar criado, utiliza-se o seu ID para criar um novo registo
para o respetivo cliente
                biblioProjetoMVVMEntities bib = new biblioProjetoMVVMEntities();
var este = bib.clientes.Where(x => cli.cc == txtNif.Text).FirstOrDefault();
                            reg.idCliente = cli.idCliente;
                            reg.idLivro = (int)combLivros.SelectedValue;
                            reg.dataInicio = DateTime.Now;
        //A função GetUsename encontra-se na ViewModelClientes - Ver anexo 2
                            reg.username = ViewModel.GetUsername();
  var livros = bib.livros.Where(x => x.idLivro == reg.idLivro).FirstOrDefault();
                            livros.estado = true;
                            onPropertyChanged("ListaClientes");
                            clienteModel.registos.Add(reg);
                            clienteModel.SaveChanges();
                            onPropertyChanged("ListaClientes");
                            bib.SaveChanges();
                            MessageBox.Show("sucesso");
                            criarClientes abc = new criarClientes();
                            abc.Show();
                            this.OnClosingRequest();
```

```
                }
                else MessageBox.Show("Escolha um Livro");
            }
            else MessageBox.Show("O campo NIF só pode possuir números");
        }
        else  MessageBox.Show("Preencha os dados corretamente");
    }
    catch (Exception error)
    {   MessageBox.Show(error.Message);  }

}
```

Pode-se visualizar como fica a parte gráfica desta janela na figura XXIX.

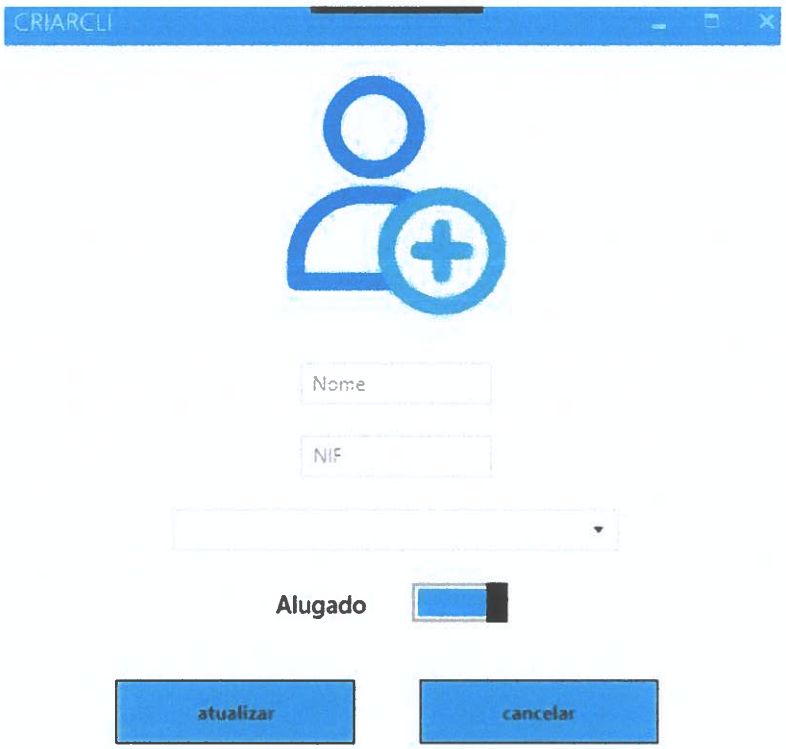

Figura 29 - Inserir Cliente
(Fonte do Autor)

Apagar Cliente

O botão responsável por apagar os clientes apenas está ativo se o funcionário estiver no grupo do AD adminAplicacao. Caso esteja no grupo Funcionarios então este botão encontra-se desativado. É no *binding* do botão que em vez de estar: *"e.CanExecute = true"*, vai estar *"e.CanExecute = vmclientes.check()"*.

Assim, na janela *"EditarClientes.xaml"*, associa-se o DataContext e o XAML desta janela ao controlador, *ViewModelClientes.cs*. De seguida faz-se o *Binding* do botão, tal como foi feito anteriormente. O processo de MVC é sempre o mesmo o que permite a reutilização de código. Passa-se para o método o cliente selecionado, o ClienteCorrente.

```
public ViewModelCliente vmclientes { get; set; }

public criarClientes()
    {
      vmclientes = (ViewModelCliente)Resources["vmclientes"];
      vmclientes.onPropertyChanged("ListaClientes");
      this.DataContext = vmclientes;
      //Evento que é disparado quando se fecha a janela
      vmclientes.ClosingRequest += (sender, e) => this.Close();
      //prenche combobox
      getCombo();

CommandBindings.Add(new CommandBinding(
      comandos.btnApagarCliente,
      (sender, e) => vmclientes.apagarCliente(vmclientes.ClienteCorrente),
       (sender, e) => e.CanExecute = vmclientes.check() ));
        }

public void getCombo()
            //Preenche a ComboBox com uma query feita em LINQ com os livros que estão
disponíveis para ser alugados
        {
            using (biblioProjetoMVVMEntities biblio = new biblioProjetoMVVMEntities())
            {
                var livrosDados = (from liv in biblio.livros
                                   where liv.estado == false
                                   select new
                                   {
                                       nomeLivro = liv.nomeLivro,
                                       idLivro = liv.idLivro,
                                   }).ToList();
                combLivrosEdit.ItemsSource = livrosDados;
            }
        }
```

Na classe comandos.cs declara-se o comando btnApagarCliente:

```
public static RoutedUICommand btnApagarCliente = new
RoutedUICommand("btnApagarCliente", "btnApagarCliente", typeof(comandos));
```

No XAML da *view*, associa-se o nome do comando ao botão:

```
<Button    Style="{StaticResource    AccentedSquareButtonStyle}"    Foreground="Black"
BorderThickness="1" FontWeight="Bold" Content="Apagar Cliente" Width="100" Height="35"
DockPanel.Dock="Right"                          Command="local:comandos.btnApagarCliente"
Margin="0,0,5,5"></Button>
```

Na *ViewModelCliente*, declara-se o método apagarCliente, que recebe o item corrente e verifica se existe na *ObservableCollection* ListaClientes e na *entitie* ClienteModel. De seguida, caso o cliente não tenha nenhum livro alugado (se limite for zero), então o cliente é apagado, caso tenha algum livro com ele, então não é apagado.

```
public void apagarCliente(cliente cli)
        {
            try
            {
                var este = ListaClientes.Where(x => x.idCliente ==
cli.idCliente).FirstOrDefault();
                if (este != null)
                {
                    este = clienteModel.clientes.Where(x => x.idCliente ==
cli.idCliente).FirstOrDefault();
                    if (este != null && este.limite == 0)
                    {
                        ListaClientes.Remove(este);
                        clienteModel.clientes.Remove(este);
                        clienteModel.SaveChanges();
                    }
                    else
                    { MessageBox.Show("Certifique-se que o cliente não tem livros
alugados");}
                }
                else return;
            }
            catch (Exception erro)
            { MessageBox.Show(erro.Message);}
        }
```

Ainda na *ViewModelCliente*, a função responsável por verificar se o botão apagar está ativo ou não é a função check(), que obtêm o *username* do utilizador autenticado no Windows através da função *GetUsername(),* utilizada no login, e como a função retorna DOMINIO*username*, é necessário separar o *username* depois do carater "\". De seguida é chamada a função *IsInGroup*, que verifica se o *username* se encontra no grupo indicado. Caso esteja retorna *True*, senão retorna *False*. A função *IsInGroup* também é utilizada no Login.

```
public Boolean check()
      {
         string nome = GetUsername();
         string[] words = nome.Split('\\');
         if (IsInGroup(words[1], "adminAplicacao"))
         { return true;}
         else { return false; }       }


public static string GetUsername()
      {
        WindowsIdentity currentUser = WindowsIdentity.GetCurrent();
         WindowsPrincipal                    winFuncionario                    =new
WindowsPrincipal(WindowsIdentity.GetCurrent());
         return winFuncionario.Identity.Name;
      }
```

Retorna *True* caso esteja no grupo e *False* caso não esteja

```
public static bool IsInGroup(string user, string group)
      {
         using (var identity = new WindowsIdentity(user))
         {
            var principal = new WindowsPrincipal(identity);
            return principal.IsInRole(group);
         }
      }
```

Pode-se visualizar a janela EditarClientes na figura abaixo.

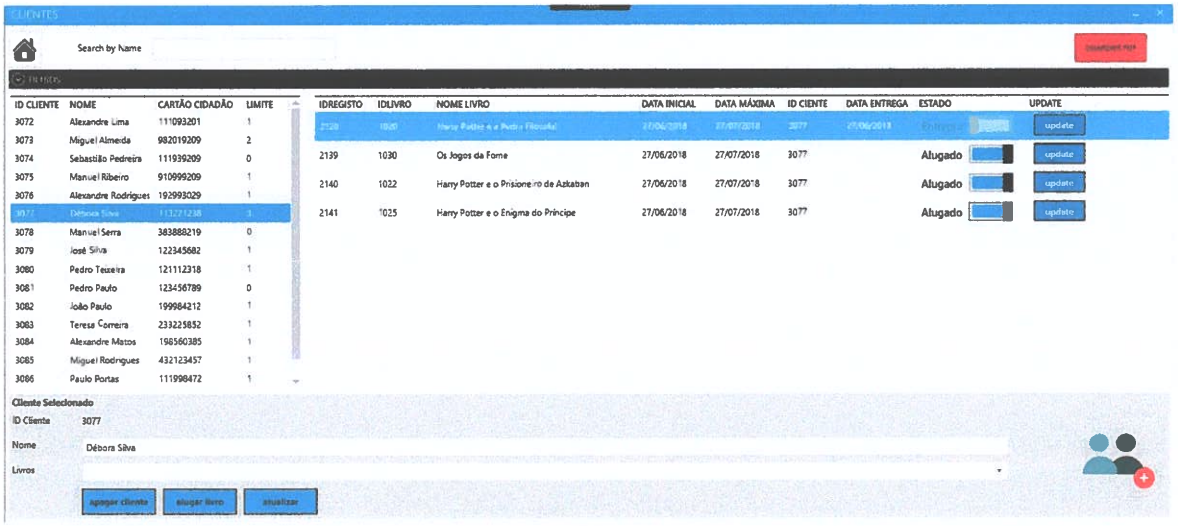

Figura 30 - Janela EditarClientes
(Fonte do Autor)

Ainda na janela EditarClientes.xaml, o botão responsável por controlar se o cliente entregou ou não um livro encontra-se dentro de uma *ListView* secundária. Isto é, esta segunda *ListView* aparece quando se seleciona um item da *ListView* principal. A lista principal mostra os clientes existentes e a lista secundária mostra os registos (livros que se encontram alugados ou que já foram entregues) de cada cliente. Para mostrar os dados existentes basta fazer um *Binding* com a *ObservableCollection* dos respetivos campos. Pode-se ver o código destas *ListViews* abaixo:

A linha que se encontra dentro do Window.Resources é obrigatória para se conseguir ligar o XAML com a ViewModelClientes.

```
<Window.Resources>
        <local:ViewModelCliente x:Key="vmclientes"></local:ViewModelCliente>
    </Window.Resources>
```

Lista principal

```
<ListView   Name="lst"   ItemsSource="{Binding ListaClientes}" Margin="3,10,0,1"
IsSynchronizedWithCurrentItem="True" VerticalAlignment="Stretch"
HorizontalAlignment="Stretch">
                <ListView.GroupStyle>
                    <GroupStyle>
                        <GroupStyle.HeaderTemplate>
                            <DataTemplate>
                                <TextBlock Text="Filtrado por número de livros por
cliente" FontWeight="Bold"/>
                            </DataTemplate>
                        </GroupStyle.HeaderTemplate>
                    </GroupStyle>
                </ListView.GroupStyle>
                <ListView.View>
                    <GridView>
                        <GridViewColumn Header="ID Cliente" Width="75"
DisplayMemberBinding="{Binding idCliente}" />
                        <GridViewColumn Header="Nome" DisplayMemberBinding="{Binding
nome}" Width="75"/>
                        <GridViewColumn Header="Cartão Cidadão"
DisplayMemberBinding="{Binding cc}" Width="120"></GridViewColumn>
                        <GridViewColumn Header="Limite" DisplayMemberBinding="{Binding
limite}" Width="60"></GridViewColumn>
                    </GridView>
                </ListView.View>
            </ListView>
```

Lista secundária

```
            <ListView  IsSynchronizedWithCurrentItem="True"

                  ItemsSource="{Binding SelectedItem.registos, ElementName=lst}"
                  x:Name="lstReg" Margin="20,10,0,0"  >

                <ListView.View>

                    <GridView>
```

```
                        <GridViewColumn Header="IdRegisto"
DisplayMemberBinding="{Binding idRegisto}"  Width="80"/>
                        <GridViewColumn Header="IdLivro"
DisplayMemberBinding="{Binding idLivro}" Width="80"/>
                        <GridViewColumn Header="Nome Livro"
DisplayMemberBinding="{Binding livro.nomeLivro}" Width="280"/>
                        <GridViewColumn Header="Data Inicial"
DisplayMemberBinding="{Binding dataInicio, StringFormat={}{0:dd/MM/yyyy}}" Width="100"
/>
                        <GridViewColumn Header="Data Máxima"
DisplayMemberBinding="{Binding dataFim,  StringFormat={}{0:dd/MM/yyyy}}" Width="100"/>
                        <GridViewColumn Header="ID Ciente"
DisplayMemberBinding="{Binding idCliente}" Width="80" />
                        <GridViewColumn Header="Data Entrega"
DisplayMemberBinding="{Binding dataEntrega,  StringFormat={}{0:dd/MM/yyyy}}"
Width="100" />

                        <GridViewColumn Header="Estado">
                            <GridViewColumn.CellTemplate>
                                <DataTemplate>
                                    <StackPanel Margin="3,2,6,2">
                                        <Controls:ToggleSwitch OnLabel="Alugado"
OffLabel="Entregue" IsChecked="{Binding entregar }" IsEnabled="{Binding entregar}"
Name="toggleEntrega"/>
                                    </StackPanel>
                                </DataTemplate>
                            </GridViewColumn.CellTemplate>


                        </GridViewColumn>
                        <GridViewColumn Header="update">
                            <GridViewColumn.CellTemplate>
                                <DataTemplate>
                                    <StackPanel Margin="6,2,6,2">
                                        <Button Style="{StaticResource
AccentedSquareButtonStyle}" Foreground="White" BorderThickness="1"  Content="Update"
Name="updatebtn" Width="70" Background="SteelBlue"
Command="local:comandos.btnEntregarListView"   />
                                    </StackPanel>
                                </DataTemplate>
                            </GridViewColumn.CellTemplate>
                        </GridViewColumn>
                    </GridView>
                </ListView.View>
            </ListView>
```

Este botão, tal como todos os outros, tem que se fazer o *Binding* senão o botão vai ficar permanentemente desativado. Vai ser passado para a ViewModelClientes a lista secundária. Para tal:

```
CommandBindings.Add(new CommandBinding(
comandos.btnEntregarListView,
(sender, e) => vmclientes.entregar(lstReg),
(sender, e) => e.CanExecute = true ));
```

Em comandos.cs é declarado o comando que vai ligar a ViewModelClientes à *view* EditarClientes.xaml.

```
public static RoutedUICommand btnEntregarListView = new
RoutedUICommand("btnEntregarListView", "btnEntregarListView", typeof(comandos));
```

Para ligar o botão ao comando, no XAML da view tem que se associar à propriedade Command o comando que se pretende, conforme se pode verificar nas linhas abaixo.

```
<Button Style="{StaticResource AccentedSquareButtonStyle}" Foreground="White"
BorderThickness="1" Content="Update" Name="updatebtn" Width="70"
Background="SteelBlue" Command="local:comandos.btnEntregarListView" />
```

Na ViewModelClientes é declarado o método "entregar" que recebe uma *ListView*. É passado o item atual da lista "lst.registos", onde se vai comparar com os dados existentes de forma a confirmar que o registo selecionado é do respetivo cliente, isto porque cada cliente pode ter até três livros. É feita uma validação de forma a garantir que antes de ser pressionado o botão para entrega do livro, é necessário alterar o botão *toggle*, que coloca a variável entrega (da tabela Registos) a *False*. Apenas depois se deve carregar no botão de entrega. Isto porque se fosse tudo feito no *toggle*, poderia haver enganos, assim primeiro é necessário carregar no *toggle* e caso não se carregue no botão de entrega então os dados não são guardados. Apenas são guardados quando se carrega primeiro no Toggle, colocando a variável que indica se o livro está alugado a *False* e depois no botão que realiza a entrega.

Desta forma, caso estas condições se confirmem, é retirado uma unidade à variável limite (visto que foi feita a entrega), o estado do registo fica a false (variável entregar, controlada pelo *toggle*), é adicionado o *username* do responsável pela entrega, adiciona-se a data de entrega e altera-se a disponibilidade do livro para False uma vez que foi entregue.

```
public void entregar(System.Windows.Controls.ListView lst)
{
    try
    {
        using (bd = new biblioProjetoMVVMEntities())
        {
            var reg = ((registo)lst.SelectedItem);
            var row = bd.registos.Where(x => x.idRegisto == reg.idRegisto && x.idCliente
== reg.idCliente).FirstOrDefault();
            if (row != null)
            {
                if (reg.entregar == false && reg.dataEntrega == null)
                {
                    reg.dataEntrega = DateTime.Now;
                    if (reg.cliente.limite > 0)
                        reg.cliente.limite = row.cliente.limite - 1;
```

```
                reg.livro.estado = false;
                reg.usernameEntrega = GetUsername();
                bd.SaveChanges();
                clienteModel.SaveChanges();
                clienteModel.registos.Add(reg);
                onPropertyChanged("ListaClientes");
                MessageBox.Show("Entregue");
                criarClientes abc = new criarClientes();
                this.OnClosingRequest();
                abc.Show();
            }
            else {     MessageBox.Show("Selecione o estado"); }
        }
    }
}
catch (Exception error)
{ MessageBox.Show(error.Message); }
}
```

Nesta *view*, "EditarClientes.xaml" é ainda possível atualizar o nome de cada cliente e transferir os registos de cada um dos clientes de forma organizada para PDF conforme se pode verificar na figura abaixo. O código para tal encontra-se em Anexos 2 – ViewModelClientes.cs.

Registo Individual Clientes

ID cliente: 3073. Nome: Miguel Almeida

Registos Cliente

idRegisto	idLivro	Livro	dataInicio	dataFim	dataEntrega	usernam e Aluguer	Usernam e Entrega
2121	1023	Harry Potter e o Cálice de Fogo	27/06/2018	27/07/2018	Por Entregar	PROJETO\alex	Ainda não foi entregue
2125	1027	O Hipnotista	27/06/2018	27/07/2018	27/06/2018	PROJETO\alex	PROJETO\alex
2126	1035	Os Maias	27/06/2018	27/07/2018	Por Entregar	PROJETO\alex	Ainda não foi entregue
2127	1034	Livro do Desassossego	27/06/2018	27/07/2018	27/06/2018	PROJETO\alex	PROJETO\alex

Figura 31 - PDF clientes-registos
(Fonte do Autor)

Alterar Livros

É possível criar, atualizar e apagar livros. Sendo que apenas podem apagar os funcionários que tiverem permissões para tal. O código responsável por estas operações encontra-se em Anexos 3 – ViewModelLivros.cs.

Sendo que o código para inserir e apagar clientes foi explicado anteriormente, agora será explicado como se atualiza um livro de forma a evitar repetições e englobar um pouco de tudo. O código foi separado entre a "*ViewModelClientes.cs*" e a *"ViewModelLivros.cs"* para não se tornar confuso. As operações feitas sobre os livros são feitas em *"ViewModelLivros*", enquanto a outra possuí informação sobre os clientes e os registos.

Na *view "catalogo.xaml"* encontram-se numa *List* todos os livros, disponíveis ou alugados. Assim, tem de se associar o XAML da *view* à *"ViewModelLivros*" e o nome do comando ao botão.

```
<Window.Resources>
        <local:arrayimagem x:Key="conversor"></local:arrayimagem>
        <local:ViewModelLivros                x:Key="vmlivros"></local:ViewModelLivros>
</Window.Resources>

  <Button    Style="{StaticResource    AccentedSquareButtonStyle}"    Foreground="Black"
BorderThickness="2"    FontWeight="Bold"    Content="Atualizar    Livro"       Width="130"
Margin="10"  Command="local:comandos.btnAtualizaLivro"  Height="40"    />
```

No Code-Behind, associa-se o DataContext e o XAML da *view* à *"ViewModelLivros.cs*" e faz-se o *Binding* do botão com a classe *"comando.cs"*. Para tal declara-se a *ViewModelLivros* numa variável global e dentro do construtor da *view* adiciona-se o código seguinte. Passa-se para o método *"atualizaLivro"* parâmetros como o registo corrente selecionado e as caixas de texto respetivas ao nome do livro, do autor e da editora.

```
        ViewModelLivros vmLivros { get; set; }
    public catalogo()
    {
        InitializeComponent();
        //associa o xaml à ViewModelLivros
        vmLivros = (ViewModelLivros)Resources["vmlivros"];
        this.DataContext = vmLivros;

CommandBindings.Add(new CommandBinding(
        comandos.btnAtualizaLivro,
        (sender,  e)  =>  vmLivros.atualizaLivro(vmLivros.LivroCorrente,  txtNomeCriar,
txtAutorCriar, txtEditoraCriar),
```

```
(sender, e) => e.CanExecute = true));
  }
```

Na classe comandos.cs, faz-se a declaração do comando:

```
public static RoutedUICommand btnAtualizaLivro = new
RoutedUICommand("btnAtualizaLivro", "btnAtualizaLivro", typeof(comandos));
```

Na classe ViewModelLivros.cs, encontra-se o método "atualizaLivro", responsável por atualizar os dados do livro selecionado na lista. Assim, se o item selecionado for igual a um dos itens da ObservableCollection ListaLivros e as caixas de texto estiverem preenchidas, então o livro é atualizado, incluindo a capa do livro.

```
public void atualizaLivro(livro liv, TextBox txtAutor, TextBox txtNome, TextBox
txtEditora)
        {
            try
            {
                var este = ListaLivros.First(x => x.idLivro == liv.idLivro);
                if (este != null)
                {
                    if (((string.IsNullOrEmpty(txtAutor.Text)) ||
string.IsNullOrEmpty(txtNome.Text)) || string.IsNullOrEmpty(txtEditora.Text))
                        MessageBox.Show("Preencha os campos que pretende alterar. Dados
não gravados.");

                    else {
                        este.nomeLivro = liv.nomeLivro;
                        este.nomeAutor = liv.nomeAutor;
                        este.editora = liv.editora;
                        este.foto = liv.foto;
                        livroModel.SaveChanges();
                        MessageBox.Show("Livros Atualizados");
                    }
                }
            }
            catch (Exception erro)
            {MessageBox.Show(erro.Message); }
        }
```

Pode-se visualizar esta janela na figura XXXII.

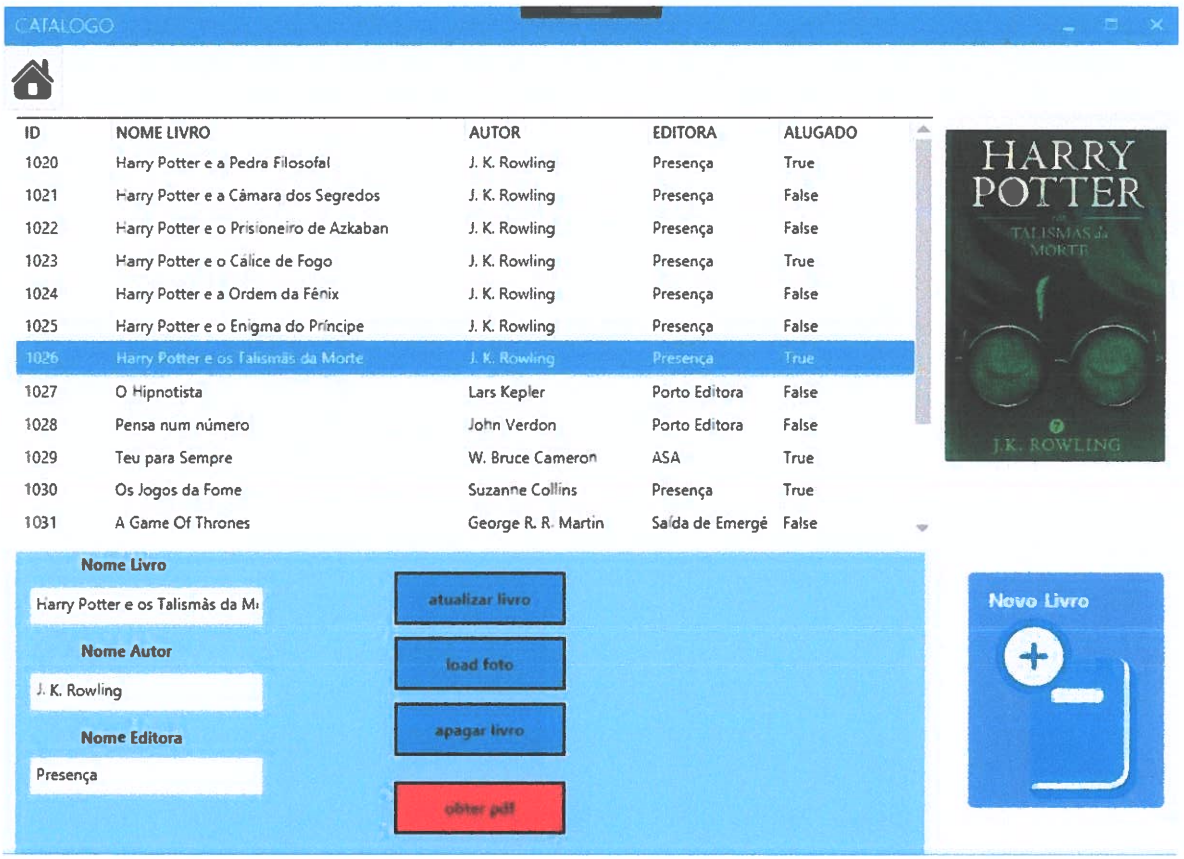

Figura 32 – Catálogo
(Fonte do Autor)

Tal como na janela que permite visualizar os registos de cada um dos clientes, bem como os clientes em si, nesta janela também é possível transferir os dados para um PDF, neste caso será dos livros existentes e se estão alugados ou disponíveis.

O código para tal encontra-se em Anexo 3 – ViewModelLivros.cs

Conclusão

Em virtude do objetivo do estudo, que consiste na implementação de uma aplicação WPF através de um ambiente virtualizado, foi criada uma infraestrutura eficiente e capaz de suportar uma aplicação que pode ser acedida por vários membros do domínio caso as suas permissões o permitam.

Devido às vantagens que a virtualização acrescenta, esta tem vindo a crescer exponencialmente em vários ramos, isto porque pode ser utilizada para acrescentar valor a uma infraestrutura física ou pode simplesmente substitui-la trazendo com isso uma diminuição nas despesas de manutenção e uma maior escalabilidade

Em função do objetivo traçado e devido aos benefícios da virtualização, conseguiu-se obter os resultados previstos uma vez que através de uma máquina física montou-se um laboratório capaz de simular a rede empresarial, com vários servidores cada um com as suas funcionalidades distintas. Além disso, através do *cluster* criado entre as máquinas obtém-se uma maior disponibilidade da aplicação. Esta aplicação será o sistema informático que uma biblioteca pode utilizar para fazer os seus registos e interações com os clientes.

Em síntese, o objetivo de estudo foi atingido visto que a infraestrutura que suporta a aplicação foi bem configurada e a aplicação foi desenvolvida de forma a fazer a gestão dos livros e dos clientes que uma biblioteca têm de fazer bem como o acesso a esta aplicação está limitado pelas permissões definidas no domínio da biblioteca.

Referências Bibliográficas

Active Directory Structure and Storage Technologies. (2014). Obtido em 2 de 5 de 2018, de docs.microsoft. disponível em:

https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc759186(v=ws.10)

Allen, R., Desmond, B., Lowe-Norris, G. & Richards, J. (2013). *Active Directory* (5ª ed.). O'Reilly.

Beaulieu, A. (2009). *Learning SQL* (2ª ed.). Sebastopol: O'Reilly.

Limeback, R. (2008). *Simply SQL*. Melbourne: sitepoint.

Marcelo Cortêz, F. L. (2013). Virtualização: Uma analise de desempenho das soluções mais utilizadas do mercado. *Infobrasil TI & Telecom*. Obtido em 2 de 27 de 2018, disponível em:
http://www.infobrasil.inf.br/userfiles/OK-Virtualiza-122050.pdf

Microsoft. (11 de April de 2013). *Windows Server 2012 Storage Virtualization Explained*. Obtido em 1 de 5 de 2018, Disponível em:

https://blogs.technet.microsoft.com/yungchou/2013/04/11/windows-server-2012-storage-virtualization-explained/

Microsoft. (2013). *Windows Server 2021 R2 - Server Virtualization Technical Overview*.

Microsoft. (29 de 11 de 2016). *Hyper-V Technology Overview*. Obtido em 2 de 5 de 2018, de Microsoft. disponível em:

https://docs.microsoft.com/pt-pt/windows-server/virtualization/hyper-v/hyper-v-technology-overview

Microsoft. (2017, b). *Active Directory Domain Services Overview*. Obtido em 8 de 5 de 2018, disponível em:

https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview

Microsoft. (2017, a). *SQL Server Management Studio (SSMS)*. Obtido em 2 de 5 de 2018, disponível em:

https://docs.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-2017

Patrick, T. (2010). *Microsoft ADO.NET 4.* Sebastopol: O'Reilly.

Peter Mell, T. G. (2011). *The Nist Definition of Cloud Computing.* Gaithersburg: U.S. Department of Commerce. Obtido em 12 de 4 de 2018, disponível em:

https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-145.pdf

Portnoy, M. (2016). *Virtualization essentials* (2ª ed.). Indianapolis: Sybex.

Richard Talaber, T. B. (2009). *Using Virtualization To Improve Data Center Efficiency.* The Green Grid.

Rosa, A. (2013). *Windows Server 2012 R2 - Curso Completo.* FCA.

Tulloch, M. (2010). *Virtualization Solutions - From the Desktop to the Datacenter.* Redmond: Microsoft.

Walat, T. (2017). *Microsoft Windows Server OS.* Obtido em 2 de 6 de 2018, Disponível em:

https://searchwindowsserver.techtarget.com/definition/Microsoft-Windows-Server-OS-operating-system

Anexos

Anexo 1 – Criação das tabelas da base de dados

Segue-se os comandos utilizados para a criação das tabelas e das respetivas relações.

Tabela livros

```
create table livros (
idLivro int primary key identity,
nomeLivro varchar(60) not null,
nomeAutor varchar (60)  not null,
editora varchar (50)  not null,
estado bit null
foto varbinary(max)
);
```

Tabela clientes

```
create table clientes(
idClienteint primary key identity,
nome varchar(50) not null ,
cc varchar(20) unique not null,
limite int null
);
```

Tabela registos

```
create table registos (
idRegisto int primary key identity,
dataInicio  date default default(GETDATE()),
dataFim as (dateadd(month,1,dataInicio)) Persisted ,
entregar bit not null,
```

```
idLivro int not null,
idCliente int not null,
username varchar(60) not null,
dataEntrega date default Getdate(),
usernameEntrega varchar(60),
Foreign key (idCliente) references clientes (idCliente)
on delete cascade on update cascade,
Foreign key (idLivro) references livros (idLivro)
on delete cascade on update cascade,
);
```

Anexo 2 – ViewModelClientes.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.ComponentModel;
using System.Collections.ObjectModel;
using System.Windows.Data;
using System.Windows.Controls;
using Microsoft.Win32;
using System.Data.Entity;
using System.Runtime.InteropServices;
using Microsoft.Win32.SafeHandles;
using System.Collections.Specialized;
using System.Windows.Media.Imaging;
using System.Windows;
using System.Data.Entity.Infrastructure;
using System.Data.Entity.Core.Objects;
using MahApps.Metro.Controls;
using MahApps.Metro.Controls.Dialogs;
using System.Security.Principal;
using System.IO;
using iTextSharp;
using iTextSharp.text;
using iTextSharp.text.pdf;
using iTextSharp.text.pdf.draw;
using System.Text.RegularExpressions;



namespace projetoMVVM
{
    public class ViewModelCliente : CloseableViewModel, INotifyPropertyChanged,
IDisposable
    {
        // Região responsável por melhorar performance em termos de memória
        #region dispose
        bool disposed = false;
        SafeHandle handle = new SafeFileHandle(IntPtr.Zero, true);
        public void Dispose()
        {
            Dispose(true);
            GC.SuppressFinalize(this);
        }

        protected virtual void Dispose(bool disposing)
        {
            if (disposed) return;
            if (disposing)
            {
                handle.Dispose();
                clienteModel = null;
                GC.Collect();
                GC.WaitForPendingFinalizers();
            }
            disposed = true;
        }
```

```
        #endregion

        #region colecções e dados
        //Criar Entitie e associar método para notificar quando há mudanças.
        biblioProjetoMVVMEntities _clienteModel;
        public biblioProjetoMVVMEntities clienteModel
        {
            get { return _clienteModel; }
            set
            {
                _clienteModel = value;
                onPropertyChanged("clienteModel");
            }
        }
        //objeto do tipo biblioProjetoVMEntities
        public biblioProjetoMVVMEntities bd;

        //Criar ObservableCollection, é public para ligar ao xml, outra vez associa-se
o método onPropertyChanged de forma a disparar o evento associado
        //quando a ObservableCollection for alterada
        public ObservableCollection<cliente> _listaClientes;
        public ObservableCollection<cliente> ListaClientes
        {
            get { return _listaClientes; }
            set
            {
                _listaClientes = value;
                onPropertyChanged("ListaClientes");
            }

        }

        //Responsáveis por saber qual é o utilizador selecionado. ex: dentro das
Listas
        public ICollectionView ViewClientes { get; set; }
        private cliente _clienteCorrente;
        public cliente ClienteCorrente
        {
            get { return _clienteCorrente; }
            set
            {
                _clienteCorrente = value;
                onPropertyChanged("ClienteCorrente");
            }
        }

        #endregion

        #region PropertyChanged

        //evento que irá ser chamado quando houver alterações, de forma a estar sempre
atualizado e andar sempre no elemento currente
        public event PropertyChangedEventHandler PropertyChanged; //evento que vai
executar
        public void onPropertyChanged(String nome)
        {
            if (PropertyChanged != null) PropertyChanged(this, new
PropertyChangedEventArgs(nome));
        }
        #endregion

        //Construtor da View
```

```
        public ViewModelCliente()
        {

            //Associa-se à ListaClientes a tabela Cliente e as tabelas Registo e Livro
por causa das relações existentes
            //Faz se a ligação à Collection
            //Associa-se o Cliente Currente ao item currente da Collection como também
as suas alterações

                clienteModel = new biblioProjetoMVVMEntities();
                ListaClientes = new
ObservableCollection<cliente>(clienteModel.clientes.Include("registos").Include("regis
tos.livro").ToList());
                ViewClientes = CollectionViewSource.GetDefaultView(ListaClientes);
                ClienteCorrente = (cliente)ViewClientes.CurrentItem;
                ViewClientes.CurrentChanged += ViewClientes_CurrentChanged;



        }

        private void ViewClientes_CurrentChanged(object sender, EventArgs e)
        {
            ClienteCorrente = (cliente)ViewClientes.CurrentItem;
        }

        #region transicoes
        //Transições entre janelas
        //OnClosingRequest é um método da classe  CloseableViewModel que irá disparar
o evento ClosingRequest, fechando assim a janela atual.
        public void criarCliView()
        {
            criarCli crCl = new criarCli();
            crCl.Show();
            this.OnClosingRequest();

        }

public void clienteGestaoView()
        {
            criarClientes crCl = new criarClientes();
            crCl.Show();
            this.OnClosingRequest();
        }
        public void catalogoView()
        {
            catalogo cat = new catalogo();
            cat.Show();
            this.OnClosingRequest();
        }
        public void voltarHome()
        {
            Home hom = new Home();
            hom.Show();
            this.OnClosingRequest();
        }
       public void cancelarBtnCliente()
        {
            criarClientes editarCli = new criarClientes();
            editarCli.Show();
            this.OnClosingRequest();
```

```
        }
        #endregion

        #region criarCliente

        //Criar Cliente
        public void insertPessoa(TextBox txtCli, TextBox txtNif, ToggleSwitch
toggSwitc, ComboBox combLivros)
        {
            try
            {
                //valida se os campos estão bem inseridos, se não é null, se o campo
NIF é constituido apenas por inteiros ou se o toggle está ativo
                if (!(string.IsNullOrEmpty(txtCli.Text) ||
string.IsNullOrEmpty(txtNif.Text))  && toggSwitc.IsChecked == true)
                {
                    int nif;
                    if (int.TryParse(txtNif.Text, out nif))
                    {
                        //verifica se a comboBox tem algum valor selecionado
                        if (combLivros.SelectedValue != null)
                        {
                            //Criar duas instancias do tipo cliente e do tipo registo
e colocando os valores inseridos nas respetivas variaveis
                            //Vão ser inseridos o idCliente (automatico), nome, NIF,
limite na tabela cliente.
                            //Na tabela registo: idRegisto(automatico), idCliente,
dataInicio (que vai gerar a dataFim, campo calculado que conta um mês depois a esta
data inicio), username,
                            // e a variavel entregar, que verifica se o livro está com
o cliente ou com a biblioteca
                            cliente cli = new cliente();
                            registo reg = new registo();
                            cli.nome = txtCli.Text;
                            cli.cc = txtNif.Text;
                            cli.limite = 1;
                            clienteModel.clientes.Add(cli);
                            clienteModel.SaveChanges();
                            ListaClientes.Add(cli);
                            //Como é necessário associar um cliente a um registo,
primeiro é criado o cliente e de seguida é utilizado
                            //o respetivo idCliente para inserir na tabela Registos o
registo do respetivo cliente
                            biblioProjetoMVVMEntities bib = new
biblioProjetoMVVMEntities();
                            var este = bib.clientes.Where(x => cli.cc ==
txtNif.Text).FirstOrDefault();
                            reg.idCliente = cli.idCliente;
                            reg.idLivro = (int)combLivros.SelectedValue;
                            reg.dataInicio = DateTime.Now;
                            reg.username = ViewModel.GetUsername();
                            //A tabela registo tem um campo para saber quando o
utilizador já entregou o respetivo livro, quando está na posse do cliente fica True
                            reg.entregar = true;
                            //alterar disponibilidade livro, visto que já não está
disponível é necessário colocar o livro do respetivo registo como ocupado, no caso
fica True
                            var livros = bib.livros.Where(x => x.idLivro ==
reg.idLivro).FirstOrDefault();
                            livros.estado = true;
                            onPropertyChanged("ListaClientes");
                            //Gravar
```

```
                            clienteModel.registos.Add(reg);
                            clienteModel.SaveChanges();
                            onPropertyChanged("ListaClientes");
                            bib.SaveChanges();
                            MessageBox.Show("sucesso");
                            criarClientes abc = new criarClientes();
                            abc.Show();
                            this.OnClosingRequest();
                        }
                        else MessageBox.Show("Escolha um Livro");
                    }
                    else MessageBox.Show("O campo NIF só pode possuir números");
                }
                else  MessageBox.Show("Preencha os dados corretamente");
            }
            catch (Exception error)
            {   MessageBox.Show(error.Message);  }

        }

        #endregion
        //Esta região irá falar de todas as operações que acontecem em
EditarClientes.xaml
        #region editarClientes
        public void entregar(System.Windows.Controls.ListView lst)
        {
            try
            {
                using (bd = new biblioProjetoMVVMEntities())
                {
                    //É passado o item Atual da lista lst.registos, onde se vai
comparar com o que existe para se encontrar qual é o registo selecionado do respetivo
cliente
                    //Isto porque cada cliente pode ter 3 registos
                    var reg = ((registo)lst.SelectedItem);
                    var row = bd.registos.Where(x => x.idRegisto == reg.idRegisto &&
x.idCliente == reg.idCliente).FirstOrDefault();
                    if (row != null)
                    {
                        //Se o ToggleButton tiver sido pressionado e nao houver uma
data de entrega, então é diminuido 1 ao limite do cliente (nº de livros)
                        // O estado do registo fica false e o estado do livro também,
é adicionada a data de entrega e o username do funcionario que fez a entrega
                        if (reg.entregar == false && reg.dataEntrega == null)
                        {
                            reg.dataEntrega = DateTime.Now;
                            if (reg.cliente.limite > 0)
                                reg.cliente.limite = row.cliente.limite - 1;
                            reg.livro.estado = false;
                            reg.usernameEntrega = GetUsername();
                            bd.SaveChanges();
                            clienteModel.SaveChanges();
                            clienteModel.registos.Add(reg);
                            onPropertyChanged("ListaClientes");
                            MessageBox.Show("Entregue");
                            criarClientes abc = new criarClientes();
                            this.OnClosingRequest();
                            abc.Show();
                        }
                        else {      MessageBox.Show("Selecione o estado"); }
                    }
                }
```

```
            }
            catch (Exception error)
            { MessageBox.Show(error.Message); }
        } //Responsável por atualizar os dados do Cliente selecionado na Lista
        public void atualizaCliente(cliente cli, TextBox txtnome)
        {
            try
            {
                //verifica se existe o cliente selecionado na ObservableCollection e
se a caixa de texto estiver preenchida o nome é alterado
                var este = ListaClientes.First(x => x.idCliente == cli.idCliente);
                if (este != null)
                {
                    if (string.IsNullOrEmpty(txtnome.Text))
                        MessageBox.Show("Prencha o campo que pretende alterar. Dados
não gravados.");
                    else
                    {
                        este.nome = cli.nome;
                        clienteModel.SaveChanges();
                        MessageBox.Show("Registo Atualizado");
                    }
                }
            }
            catch (Exception erro)
            { MessageBox.Show(erro.Message); }
        }

        //Apagar Cliente, só poderá ser executado pelos funcionarios que pertencem ao
grupo adminAplicacao (está disable para todos os outros)
        public void apagarCliente(cliente cli)
        {
            try
            {   //Caso o utilizador exista, só pode ser apagado se não tiver nenhum
livro ativo (Se Limite for igual a 0)
                var este = ListaClientes.Where(x => x.idCliente ==
cli.idCliente).FirstOrDefault();
                if (este != null)
                {
                    este = clienteModel.clientes.Where(x => x.idCliente ==
cli.idCliente).FirstOrDefault();
                    if (este != null && este.limite == 0)
                    {
                        ListaClientes.Remove(este);
                        clienteModel.clientes.Remove(este);
                        clienteModel.SaveChanges();
                    }
                    else
                    { MessageBox.Show("Certifique-se que o cliente não tem livros
alugados");}
                }
                else return;

            }
            catch (Exception erro)
            { MessageBox.Show(erro.Message);}
        }



        //Alugar Livro
```

```
        public void alugarLivro(cliente cli, ComboBox combLivrosEdit,
System.Windows.Controls.ListView lst)
        {
            try
            {
                using (bd = new biblioProjetoMVVMEntities())
                {
                    var este = ListaClientes.First(x => x.idCliente == cli.idCliente);
                    if (este != null)
                    {
                        //Caso o cliente tenha três ou mais livros, não pode alugar
mais
                        if (este.limite > 2)
                            MessageBox.Show("Cada cliente apenas pode ter 3 livros");
                        else
                        {
                            //Se a combobox tiver algum valor, é criado um novo
registo através do idCliente do cliente selecionado
                            if (combLivrosEdit.SelectedValue != null)
                            {
                                registo reg = new registo();
                                reg.idCliente = este.idCliente;
                                reg.idLivro = (int)combLivrosEdit.SelectedValue;
                                reg.dataInicio = DateTime.Now;
                                reg.username = ViewModel.GetUsername();
                                reg.entregar = true;
                                este.limite = este.limite + 1;
                                bd.SaveChanges();
                                clienteModel.registos.Add(reg);
                                clienteModel.SaveChanges();
                                onPropertyChanged("ListaClientes");
                                //É necessário também alterar o estado do livro na
tabela Livros, para tal, através da relação entre as tabelas registos e livros em
idLivro
                                var livroRegisto = bd.registos.Where(x => x.idLivro ==
reg.idLivro).First();
                                livroRegisto.livro.estado = true;
                                clienteModel.SaveChanges();
                                bd.SaveChanges();
                                onPropertyChanged("ListaClientes");
                                criarClientes abc = new criarClientes();
                                this.OnClosingRequest();
                                abc.Show();
                                MessageBox.Show("Sucesso");
                            }
                        }
                    }
                }
            }
            catch (Exception error)
            {MessageBox.Show(error.Message); }
        }

        //Função que vai verificar se o Funcionario (utilizador autenticado no
Windows) pertence ao grupo de domínio adminAplicacao, se pertencer o botão
        //para apagar vai estar ativo. Outro caso irá estar desativado. Ver Code-
Behind EditarClientes.xaml
        public Boolean check()
        {
            string nome = GetUsername();
            string[] words = nome.Split('\\');
            if (IsInGroup(words[1], "adminAplicacao"))
```

```
            {  return true;}
            else {  return false;  }
        } #endregion

        #region Login


        [System.Runtime.InteropServices.DllImport("advapi32.dll")]

        public static extern bool LogonUser(string userName, string domainName, string
password, int LogonType, int LogonProvider, ref IntPtr phToken);
        //Retorna o username do funcionario com : PROJETO\username
        public static string GetUsername()
        {
            WindowsIdentity currentUser = WindowsIdentity.GetCurrent();
            WindowsPrincipal winFuncionario = new
WindowsPrincipal(WindowsIdentity.GetCurrent());
            return winFuncionario.Identity.Name;
        }
        //Verifica se o Funcionario está num determinado grupo
        public static bool IsInGroup(string user, string group)
        {
            using (var identity = new WindowsIdentity(user))
            {
                var principal = new WindowsPrincipal(identity);
                return principal.IsInRole(group);
            }
        }
        //Valida o Funcionario caso o username e a password correspondam aos dados que
foram utilizados para autenticar sessão no Windows
        private bool validacao(string userName, string password, string dominio)
        {
            IntPtr tokenHandler = IntPtr.Zero;
            bool isValid = LogonUser(userName, dominio, password, 2, 0, ref
tokenHandler);
            return isValid;
        }

        public void Login(TextBox txtUser, PasswordBox txtPW)
        {
            string dominio = "projeto";
            bool ligado = false;
            //obtem o username
            string winFuncionario = GetUsername();
            try
            {
                //Valida se os campos estão preenchidos e se corresponde à
autenticação utilizada para iniciar sessão no Windows
                if (txtUser.Text != null && txtPW.Password != null)
                {
                    if
(winFuncionario.ToLowerInvariant().Contains(txtUser.Text.Trim().ToLowerInvariant()) &&
winFuncionario.ToLowerInvariant().Contains(dominio.Trim().ToLowerInvariant()))
                    {
                        ligado = validacao(txtUser.Text.Trim(), txtPW.Password.Trim(),
dominio.Trim());
                        if (ligado)
                        {
                             string nome = ViewModelCliente.GetUsername();
                            //Separou-se porque o resultado é: PROJETO\Username. Mas a
função IsInGroup aceita apenas o username
                            string[] words = nome.Split('\\');
```

```
                            //Caso o utilizador esteja no grupo adminAplicacao
                            if (ViewModelCliente.IsInGroup(words[1],
"adminAplicacao"))
                            {
                                MessageBox.Show("Bem vindo: "+ words[1] + " do Grupo:
adminAplicacao" );
                                Home paginaNova = new Home();
                                paginaNova.Show();
                                this.OnClosingRequest();
                            }
                            //Caso o utilizador esteja no grupo Funcionarios
                            else if (ViewModelCliente.IsInGroup(words[1],
"Funcionarios"))
                            {
                                MessageBox.Show("Bem vindo: " + words[1] + " do Grupo:
Funcionarios");
                                Home paginaNova = new Home();
                                paginaNova.Show();
                                this.OnClosingRequest();

                            }
                            //Outros grupos e pessoas do dominio não tem acesso à
aplicacao
                            else
                            {
                                MessageBox.Show("Nao tem acesso");

                            }
                        }
                        else
                        { MessageBox.Show("Username ou Password incorretos.", "Tenta
outra vez"); }
                    }
                    else
                    { MessageBox.Show("Username ou Password incorretos.", "Tenta outra
vez"); }
                }
                else { MessageBox.Show("Insira as suas credenciais"); return; }
            }
            catch (Exception error)
            { MessageBox.Show(error.Message); }
        }
        #endregion
        //Gerar PDF com os registos de cada Cliente
public void pdfTodo()
        {
            Document doc = new Document(PageSize.A4, 15f, 15f, 15f, 15f);
            try
            {
                //definir a pasta onde ficam os pdf. Pasta Documentos
                string caminho = Environment.CurrentDirectory;
                caminho = caminho.Substring(0, caminho.IndexOf("bin")) +
"documentos\\todos.pdf";
                FileStream fs = new FileStream(caminho, FileMode.Create,
FileAccess.Write);
                PdfWriter w = PdfWriter.GetInstance(doc, fs);
                doc.Open();
                //vai se percorrendo cada cliente e gera-se uma tabela com os seus
registos. A
                foreach (cliente cli in ListaClientes)
                {  Font titulo1 = FontFactory.GetFont("COURIER", 20f);
```

```
                    titulo1.SetStyle(Font.BOLD);
                    Paragraph p1 = new Paragraph("Registo Individual Clientes \n",
titulo1);
                    p1.Alignment = Element.ALIGN_CENTER;
                    doc.Add(p1);
                    doc.Add(new Phrase("\n\n\n"));
                    Font titulo2 = FontFactory.GetFont("COURIER", 15f);
                    Paragraph p2 = new Paragraph("ID cliente: "+
cli.idCliente.ToString() + ". Nome: " + cli.nome, titulo2);
                    p2.IndentationLeft = 120;
                    p2.SetLeading(5.0f, 4.0f);
                    doc.Add(p2);
                    doc.Add(new Phrase("\n\n"));
                    PdfPTable table = new PdfPTable(8);
                    table.DefaultCell.BorderColor = BaseColor.BLACK;
                    table.DefaultCell.Border = Rectangle.BOX;
                    table.DefaultCell.HorizontalAlignment = Element.ALIGN_CENTER;
                    PdfPCell cell = new PdfPCell(new Phrase("Registos Cliente",
titulo1));
                    cell.Colspan = 8;
                    cell.HorizontalAlignment = Element.ALIGN_CENTER;
                    cell.VerticalAlignment = Element.ALIGN_CENTER;
                    table.AddCell(cell);
                    table.AddCell(new Phrase("idRegisto"));
                    table.AddCell(new Phrase("idLivro"));
                    table.AddCell(new Phrase("Livro"));
                    table.AddCell(new Phrase("dataInicio"));
                    table.AddCell(new Phrase("dataFim"));
                    table.AddCell(new Phrase("dataEntrega"));
                    table.AddCell(new Phrase("username Aluguer"));
                    table.AddCell(new Phrase("Username Entrega"));
                    foreach (registo reg in cli.registos)
                    {
                        //insere-se os dados nas células da tabela criadas
anteriormente
                        table.AddCell(new Phrase(reg.idRegisto.ToString()));
                        table.AddCell(new Phrase(reg.idLivro.ToString()));
                        table.AddCell(new Phrase(reg.livro.nomeLivro.ToString()));
                        table.AddCell(new Phrase(reg.dataInicio.ToShortDateString()));
                        table.AddCell(new
Phrase(reg.dataFim.Value.ToShortDateString()));
                        //caso nao esteja entregue aparece: Por Entregar, caso esteja
aparece a Data de entrega
                        if (reg.dataEntrega.HasValue) { table.AddCell(new
Phrase(reg.dataEntrega.Value.ToShortDateString())); }
                        else
                            table.AddCell(new Phrase("Por Entregar"));
                        table.AddCell(new Phrase(reg.username.ToString()));
                        if (reg.usernameEntrega != null)
                        {table.AddCell(new Phrase(reg.usernameEntrega.ToString()));}
                        else  table.AddCell(new Phrase("Ainda não foi entregue"));
                    }
                    doc.Add(table);
                    doc.NewPage();
                }//foreach clientes
                MessageBox.Show("Sucesso, PDF guardado na pasta");
                doc.Close();
            }
            catch (Exception error)
            { MessageBox.Show(error.Message); }
        }
```

```
#region filter
        //Métodos gerado para procurar à medida que se escreve na Lista
(EditarClientes.xaml).
        private string _procurar;
        public string Procurar
        {
            get { return _procurar; }
            set
            {
                _procurar = value;
                ViewClientes.Filter = FiltrarData;
                onPropertyChanged("Procurar");
            }
        }
        private bool FiltrarData(object item)
        {
            var value = (cliente)item;
            if (value == null || value.nome == null)
                return false;

            return value.nome.Contains(Procurar);
        }

        //Metodos utilizados para ordenar pelo idCliente e pelo número de livros por
utilizador
        private bool _groupByLimite;
        public bool GroupByLimite
        {
            get { return _groupByLimite; }
            set
            {
                _groupByLimite = value;
                GroupData();
                onPropertyChanged("GroupByLimite");
            }
        }
        private void GroupData()
        {
            using (ViewClientes.DeferRefresh())
            {
                ViewClientes.GroupDescriptions.Clear();
                if (_groupByLimite) ViewClientes.GroupDescriptions.Add(new
PropertyGroupDescription("limite"));
            }
        }

        private bool _directionAscending = false;
        public bool DirectionAscending
        {
            get { return _directionAscending; }
            set
            {
                _directionAscending = value;
                sortData();
                onPropertyChanged("DirectionAscending");
            }

        }
        private bool _sortById = false;
        public bool SortById
        {
            get { return _sortById; }
```

```
            set
            {
                _sortById = value;
                sortData();
                onPropertyChanged("SortById");
            }
        }

        public void sortData()
        {
            ListSortDirection direccao = (_directionAscending) ?
ListSortDirection.Ascending : ListSortDirection.Descending;
            using (ViewClientes.DeferRefresh())
            {
                ViewClientes.SortDescriptions.Clear();
                if (_sortById) ViewClientes.SortDescriptions.Add(new
SortDescription("idCliente", direccao));
            }
        }

        #endregion


    }
}
```

Anexo 3 – ViewModelLivros.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.ComponentModel;
using System.Collections.ObjectModel;
using System.Windows.Data;
using System.Windows.Controls;
using Microsoft.Win32;
using System.Data.Entity;
using System.Runtime.InteropServices;
using Microsoft.Win32.SafeHandles;
using System.Collections.Specialized;
using System.Windows.Media.Imaging;
using System.Windows;
using System.Data.Entity.Infrastructure;
using System.Data.Entity.Core.Objects;
using iTextSharp.text;
using iTextSharp.text.pdf;
using iTextSharp.text.pdf.draw;
using System.IO;
using System.Text.RegularExpressions;



namespace projetoMVVM
{
   public class ViewModelLivros : CloseableViewModel, INotifyPropertyChanged,
IDisposable
    {

        // Região responsável por melhorar performance em termos de memória
        #region dispose
        bool disposed = false;
        SafeHandle handle = new SafeFileHandle(IntPtr.Zero, true);

        public void Dispose()
        {
            Dispose(true);
            GC.SuppressFinalize(this);
        }

        protected virtual void Dispose(bool disposing)
        {
            if (disposed) return;

            if (disposing)
            {
                handle.Dispose();
                livroModel = null;
                GC.Collect();
                GC.WaitForPendingFinalizers();
            }
            disposed = true;
        }
```

```
        #endregion

        #region colecções e dados

        //Criar Entitie e o método para notificar a Entitie quando há mudanças.
        biblioProjetoMVVMEntities _livroModel;
        public biblioProjetoMVVMEntities livroModel
        {
            get { return _livroModel; }
            set
            {
                _livroModel = value;
                onPropertyChanged("livroModel");
            }
        }

        //Criar ObservableCollection, é public para ligar ao xml, outra vez associa-se
o método onPropertyChanged de forma a disparar o evento associado
        //quando a ObservableCollection for alterada
        public ObservableCollection<livro> _listaLivros;
        public ObservableCollection<livro> ListaLivros
        {
            get { return _listaLivros; }
            set
            {
                _listaLivros = value;
                onPropertyChanged("ListaLivros");
            }
        }


        #endregion



  #region PropertyChanged
        //evento que irá notificar a classe, de forma a estar sempre atualizado quando
ha alterações
        public event PropertyChangedEventHandler PropertyChanged; //evento que vai
executar
        public void onPropertyChanged(String nome)
        {
            if (PropertyChanged != null) PropertyChanged(this, new
PropertyChangedEventArgs(nome));
        }
        #endregion

        //Responsáveis por saber qual é o utilizador selecionado ou corrente
        public ICollectionView ViewLivros { get; set; }
        private livro _LivroCorrente { get; set; }
        public livro LivroCorrente
        {
            get { return _LivroCorrente; }
            set
            {
                _LivroCorrente = value;
               onPropertyChanged("LivroCorrente");
            }
        }

        public ViewModelLivros() //Construtor
```

```
        {
            //Associa-se à ListaLivros a tabela Livros e as tabelas Registo e Clientes
devido às relações existentes
            //Faz se a ligação à Collection
            //Associa-se o Livro Currente ao item currente da Collection como também
as suas alterações

            livroModel = new biblioProjetoMVVMEntities();
                ListaLivros = new
ObservableCollection<livro>(livroModel.livros.Include("registos").Include("registos.cl
iente").ToList());
                ViewLivros = CollectionViewSource.GetDefaultView(ListaLivros);
                ViewLivros.CurrentChanging += ViewLivros_CurrentChanging;
                LivroCorrente = (livro)ViewLivros.CurrentItem ;
                ViewLivros.CurrentChanged += ViewLivros_CurrentChanged;
        }

        private void ViewLivros_CurrentChanged(object sender, EventArgs e)
        {
            LivroCorrente = (livro)ViewLivros.CurrentItem;
        }
        private void ViewLivros_CurrentChanging(object sender,
CurrentChangingEventArgs e)
        {
            LivroCorrente =(livro) ViewLivros.CurrentItem ;
        }


        #region criarLivros
        //Inserir Livro
        public void insertLivro (TextBox txtNome, TextBox txtAutor, TextBox
txtEditora, System.Windows.Controls.Image img)
        {
            try
            {
                //se as caixas de texto não estiverem vazias, então é inserido um novo
Livro
                if (!( string.IsNullOrEmpty(txtNome.Text) ||
string.IsNullOrEmpty(txtAutor.Text) || string.IsNullOrEmpty(txtEditora.Text)))
                {
                   // constituido por idLivro(automatico), nome , autor, editora, foto
e estado (se está alugado ou não)
                    byte[] foto = utils.ConvertBitmapSourceToByteArray(img.Source);
                    livro liv = new livro();
                    liv.nomeLivro = txtNome.Text;
                    liv.nomeAutor = txtAutor.Text;
                    liv.editora = txtEditora.Text;
                    liv.estado = false;
                    if (foto != null) liv.foto = foto;
                    livroModel.livros.Add(liv);
                    livroModel.SaveChanges();
                    ListaLivros.Add(liv);
                    onPropertyChanged("ListaLivros");
                    MessageBox.Show("Livro Inserido");
                    catalogo catalog = new catalogo();
                    catalog.Show();
                    this.OnClosingRequest();
                }
                else MessageBox.Show("Preencha os campos");
            }
            catch (Exception error)
            { MessageBox.Show(error.Message); }
```

```
        }
//Abre uma janela para selecionar uma fotografia, para a capa do livro
        public void foto_click( System.Windows.Controls.Image img)
        {
            OpenFileDialog dlg = new OpenFileDialog();
            dlg.DefaultExt = ".png";
            dlg.Filter = "*.*|*.*|JPEG Files (*.jpeg)|*.jpeg|PNG Files
(*.png)|*.png|JPG Files (*.jpg)|*.jpg|GIF Files (*.gif)|*.gif";
            dlg.Title = "Foto";
            Nullable<bool> result = dlg.ShowDialog();
            if (result == true)
            { img.Source = utils.picpelocaminho(dlg.FileName); }
        }
        //Botão cancelar em criarLivro.xaml
        public void cancelLivro()
        {
            catalogo catal = new catalogo();
            catal.Show();
            this.OnClosingRequest();
        }
        #endregion

        #region catalogo

        //Esta regiao é sobre Catalogo.xaml
        //Responsável por atualizar os dados do Livro selecionado na Lista

        public void atualizaLivro(livro liv, TextBox txtAutor, TextBox txtNome,
TextBox txtEditora)
        {
            try
            {
                //se o Item selecionado for igual a um dos itens da
ObservableCollection e as caixas de texto estiverem prenchidas
                //é atualizado o livro
                var este = ListaLivros.First(x => x.idLivro == liv.idLivro);
                if (este != null)
                {
                    if (((string.IsNullOrEmpty(txtAutor.Text)) ||
string.IsNullOrEmpty(txtNome.Text)) || string.IsNullOrEmpty(txtEditora.Text))
                        MessageBox.Show("Preencha os campos que pretende alterar.
Dados não gravados.");

                    else {
                        este.nomeLivro = liv.nomeLivro;
                        este.nomeAutor = liv.nomeAutor;
                        este.editora = liv.editora;
                        este.foto = liv.foto;
                        livroModel.SaveChanges();
                        MessageBox.Show("Livros Atualizados");
                    }
                }
            }
            catch (Exception erro)
            {MessageBox.Show(erro.Message); }
        }

        //Apagar livro, só poderá ser executado pelos funcionarios que pertencem ao
grupo adminAplicacao (está disable para todos os outros)
        public void apagarLivro(livro liv)
        {
```

```
            try
            {
                //Caso o livro seleciondo exista na ObservableCollection, só pode ser
apagado se não estiver alugado (ou seja, apenas pode ser apagado se o estado=False)
                var este = ListaLivros.Where(x => x.idLivro ==
liv.idLivro).FirstOrDefault();
                if (este != null)
                {
                    este = livroModel.livros.Where(x => x.idLivro ==
liv.idLivro).FirstOrDefault();
                    if (este != null && este.estado == false)
                    {
                        ListaLivros.Remove(este);
                        livroModel.livros.Remove(este);
                        livroModel.SaveChanges();
                    }
                    else
                    { MessageBox.Show("Certifique-se que o livro não está alugado
antes de apagar"); }
                }
                else return;
            }
            catch (Exception erro)
            { MessageBox.Show(erro.Message); }
        }

        //Função que vai verificar se o Funcionario (utilizador autenticado no
Windows) pertence ao grupo de domínio adminAplicacao, se pertencer o botão
        //para apagar vai estar ativo. Outro caso irá estar desativado. Ver Code-
Behind catalogo.xaml
        public Boolean  checkAdmin()
        {
           string nome=  ViewModelCliente.GetUsername();
            string[] words = nome.Split('\\');
            if ( ViewModelCliente.IsInGroup(words[1], "adminAplicacao"))
            {return true;}
           else
             { return false;}
        }
        #endregion

        #region Transicoes
        //Transições
        public void voltarHome()
        {
            Home hom = new Home();
            hom.Show();
            this.OnClosingRequest();
        }

        public void viewCriarLivro()
        {
            criarLivro crCl = new criarLivro();
            this.OnClosingRequest();

            crCl.Show();
        }
        #endregion
```

```
public void pdfLivro()
        {
            Document doc = new Document(PageSize.A4, 15f, 15f, 15f, 15f);
            try
            {
                //definir a pasta onde ficam os pdf. Pasta Documentos
                string caminho = Environment.CurrentDirectory;
                caminho = caminho.Substring(0, caminho.IndexOf("bin")) +
"documentos\\livros.pdf";
                FileStream fs = new FileStream(caminho, FileMode.Create,
FileAccess.Write);
                PdfWriter w = PdfWriter.GetInstance(doc, fs);
                doc.Open();
                //vai se percorrendo cada cliente e gera-se uma tabela com os seus
registos. A
                foreach (livro liv in ListaLivros)
                {
                    Font titulo1 = FontFactory.GetFont("COURIER", 20f);
                    titulo1.SetStyle(Font.BOLD);
                    Paragraph p1 = new Paragraph("Dados de cada livro  \n", titulo1);
                    p1.Alignment = Element.ALIGN_CENTER;
                    doc.Add(p1);
                    doc.Add(new Phrase("\n\n\n"));
                    Font titulo2 = FontFactory.GetFont("COURIER", 15f);

                    Paragraph p2 = new Paragraph("ID Livro: " +
liv.idLivro.ToString(), titulo2);
                    Paragraph p3 = new Paragraph("Nome Livro: " + liv.nomeLivro,
titulo2);
                    Paragraph p4 = new Paragraph("Nome Autor: " + liv.nomeAutor,
titulo2);
                    Paragraph p5 = new Paragraph("Nome Editora: " + liv.editora,
titulo2);
                    Paragraph p6 = new Paragraph("Alugado: " + liv.estado, titulo2);
                    p2.IndentationLeft = 120;
                    p2.SetLeading(5.0f, 4.0f);
                    p3.IndentationLeft = 120;
                    p3.SetLeading(5.0f, 4.0f);
                    p4.IndentationLeft = 120;
                    p4.SetLeading(5.0f, 4.0f);
                    p5.IndentationLeft = 120;
                    p5.SetLeading(5.0f, 4.0f);
                    p6.IndentationLeft = 120;
                    p6.SetLeading(5.0f, 4.0f);
                    doc.Add(p2);
                    doc.Add(p3);
                    doc.Add(p4);
                    doc.Add(p5);
                    doc.Add(p6);
                    doc.Add(new Phrase("\n\n"));

                    doc.NewPage();
                }//foreach livros
                MessageBox.Show("Sucesso, PDF guardado na pasta");
                doc.Close();
            }

            catch (Exception error)
            { MessageBox.Show(error.Message);}

        }
    } }
```

Anexo 4 – comandos.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Input;
using System.Windows.Controls;

namespace projetoMVVM
{
  public static  class comandos
    {
        //login
        public static RoutedUICommand login = new RoutedUICommand("login", "login",
typeof(comandos));
        //navegar
        public static RoutedUICommand catalogoMenu = new
RoutedUICommand("catalogoMenu", "catalogoMenu", typeof(comandos));
        public static RoutedUICommand clientesGestao = new
RoutedUICommand("clientesGestao", "clientesGestao", typeof(comandos));
        public static RoutedUICommand voltar = new RoutedUICommand("voltar", "voltar",
typeof(comandos));
        public static RoutedUICommand botaoCriarMudar = new
RoutedUICommand("novaTransaçãoMenu", "novaTransaçãoMenu", typeof(comandos));
        //Criar Cliente
        public static RoutedUICommand insertCliente = new
RoutedUICommand("insertCliente", "insertCliente", typeof(comandos));
        public static RoutedUICommand btnCancelarCliente = new
RoutedUICommand("btnCancelarCliente", "btnCancelarCliente", typeof(comandos)); //FALTA
POR
        //Editar Clientes
        public static RoutedUICommand btnEntregarListView = new
RoutedUICommand("btnEntregarListView", "btnEntregarListView", typeof(comandos));
        public static RoutedUICommand pdfTodos = new RoutedUICommand("pdfTodos",
"pdfTodos", typeof(comandos));
        public static RoutedUICommand btnApagarCliente = new
RoutedUICommand("btnApagarCliente", "btnApagarCliente", typeof(comandos));
        public static RoutedUICommand btnAlugarLivro = new
RoutedUICommand("btnAlugarLivro", "btnAlugarLivro", typeof(comandos));
        public static RoutedUICommand btnAtualizaCliente = new
RoutedUICommand("btnAtualizaCliente", "btnAtualizaCliente", typeof(comandos));
        //Inserir Livro
        public static RoutedUICommand btnInserirLivro = new
RoutedUICommand("btnInserirLivro", "btnInserirLivro", typeof(comandos));
        public static RoutedUICommand foto_click = new
RoutedUICommand("guardaFotopath", "guardaFotopath", typeof(comandos));
        public static RoutedUICommand cancelarLivro = new
RoutedUICommand("cancelarLivro", "cancelarLivro", typeof(comandos));
        //Catalogo
        public static RoutedUICommand btnInserirLivroView = new
RoutedUICommand("btnInserirLivroView", "btnInserirLivroView", typeof(comandos));
        public static RoutedUICommand btnAtualizaLivro = new
RoutedUICommand("btnAtualizaLivro", "btnAtualizaLivro", typeof(comandos));
        public static RoutedUICommand btnapagarLivro = new
RoutedUICommand("btnapagarLivro", "btnapagarLivro", typeof(comandos));
        public static RoutedUICommand btnPDF = new RoutedUICommand("btnPDF", "btnPDF",
typeof(comandos)); }}
```

Anexo 5 – Login.xaml

Código em XAML:

```
<Controls:MetroWindow x:Class="projetoMVVM.Login"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:projetoMVVM"
        xmlns:Controls="clr-namespace:MahApps.Metro.Controls;assembly=MahApps.Metro"
        xmlns:Dialog="clr-
namespace:MahApps.Metro.Controls.Dialogs;assembly=MahApps.Metro"
        mc:Ignorable="d"
        Title="Login" Height="400" Width="500" WindowStartupLocation="CenterScreen">
    <Window.Resources>
        <local:ViewModelCliente x:Key="nomes"></local:ViewModelCliente>
    </Window.Resources>
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="160"></RowDefinition>
            <RowDefinition></RowDefinition>
        </Grid.RowDefinitions>
        <Image Source="imagens/lock.png" Margin="20"></Image>
        <StackPanel Grid.Row="1">
            <TextBox Controls:TextBoxHelper.Watermark="Username" Name="txtUser"
HorizontalAlignment="Center" Height="32" Margin="10" TextWrapping="Wrap"
VerticalAlignment="Center" Width="150"/>
        <PasswordBox Controls:PasswordBoxHelper.CapsLockWarningToolTip="CapsLock
ligado" Name="txtPW" Width="150" Height="28" Margin="10" VerticalAlignment="Center"
HorizontalAlignment="Center" ></PasswordBox>
            <Button  Style="{StaticResource AccentedSquareButtonStyle}"
Foreground="Black" BorderThickness="1" FontWeight="Bold" Content="Entrar"
HorizontalAlignment="Center" Margin="10" VerticalAlignment="Center" Width="184"
Command="local:comandos.login"  />
        </StackPanel>
    </Grid>
</Controls:MetroWindow>
```

Código em Code-Behind:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using MahApps.Metro.Controls;
using MahApps.Metro.Controls.Dialogs;
using System.Security.Principal;
using System.Runtime.InteropServices;


namespace projetoMVVM
{
    public partial class Login : MetroWindow
    {
        public ViewModelCliente vmclientes { get; set; }
        public Login()
        {
            InitializeComponent();
            //associa o xaml e o DataContext à vmclientes
            vmclientes = (ViewModelCliente)Resources["nomes"];
            this.DataContext = vmclientes;
            //Evento que é disparado quando se fecha a janela
            vmclientes.ClosingRequest += (sender, e) => this.Close();
            //Binding do comando e do método com o botão
            CommandBindings.Add(new CommandBinding(
              comandos.login,
              (sender, e) => vmclientes.Login(txtUser, txtPW),
               (sender, e) => e.CanExecute = true

      ));
        }
    }
}
```

Anexo 6 – Home.xaml

Código em XAML:

```
<Controls:MetroWindow  x:Class="projetoMVVM.Home"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:projetoMVVM"
        mc:Ignorable="d"
        xmlns:Controls="clr-namespace:MahApps.Metro.Controls;assembly=MahApps.Metro"
        xmlns:Dialog="clr-
namespace:MahApps.Metro.Controls.Dialogs;assembly=MahApps.Metro"

        Title="Home" Height="450" Width="800" WindowStartupLocation="CenterScreen">
    <Window.Resources>

        <local:ViewModelCliente x:Key="vmclientes"></local:ViewModelCliente>
    </Window.Resources>
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"></ColumnDefinition>
            <ColumnDefinition Width="*"></ColumnDefinition>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="auto"></RowDefinition>
            <RowDefinition></RowDefinition>
        </Grid.RowDefinitions>
        <TextBlock  Text="{Binding texto}" Width="90" Height="50"></TextBlock>
        <Button Style="{DynamicResource MetroCircleButtonStyle}" Grid.Column="0"
Grid.Row="1" Background="LightBlue"  HorizontalAlignment="Center"  Margin="40"
VerticalAlignment="Center" Width="215" Height="260"
Command="local:comandos.catalogoMenu">
            <StackPanel>
                <Image Source="imagens/bookshelf.png"  ></Image>
                <TextBlock Text="Catálogo" VerticalAlignment="Center"
HorizontalAlignment="Center" FontWeight="Bold" FontSize="25"></TextBlock>

            </StackPanel>
        </Button>
        <Button Style="{DynamicResource MetroCircleButtonStyle}"   Grid.Row="1"
Grid.Column="1" Background="LightGray" HorizontalAlignment="Center"
VerticalAlignment="Center"  Width="215" Height="260"
Command="local:comandos.clientesGestao">
            <StackPanel>
                <Image Source="imagens/gestao.png" ></Image>
                <TextBlock Text="Gestão Clientes" VerticalAlignment="Center"
HorizontalAlignment="Center" FontWeight="Bold" FontSize="25"></TextBlock>

            </StackPanel>
        </Button>

    </Grid>
</Controls:MetroWindow>
```

Código em Code-Behind:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using MahApps.Metro.Controls;
using MahApps.Metro.Controls.Dialogs;

namespace projetoMVVM
{
    public partial class Home : MetroWindow
    {
        public ViewModelCliente vmclientes { get; set; }
        public Home()
        {
            InitializeComponent();
            //associa o xaml e o DataContext à vmclientes
            vmclientes = (ViewModelCliente)Resources["vmclientes"];
            this.DataContext = vmclientes;
            //Evento que é disparado quando se fecha a janela
            vmclientes.ClosingRequest += (sender, e) => this.Close();

            //Binding do comando e do método com o botão (xaml)
            CommandBindings.Add(new CommandBinding(
             comandos.clientesGestao,
             (sender, e) => vmclientes.clienteGestaoView(),
              (sender, e) => e.CanExecute = true     ));

            CommandBindings.Add(new CommandBinding(
            comandos.catalogoMenu,
            (sender, e) => vmclientes.catalogoView(),
             (sender, e) => e.CanExecute = true ));
        }
    }
}
```

Anexo 7 – EditarClientes.xaml

Código em XAML:

```
<Controls:MetroWindow x:Class="projetoMVVM.criarClientes"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:projetoMVVM"
        mc:Ignorable="d"
        xmlns:Controls="clr-namespace:MahApps.Metro.Controls;assembly=MahApps.Metro"
        xmlns:Dialog="clr-
namespace:MahApps.Metro.Controls.Dialogs;assembly=MahApps.Metro"
        Title="Clientes" Height="769" Width="800" WindowState="Maximized"
ResizeMode="CanMinimize">
    <Window.Resources>

        <local:ViewModelCliente x:Key="vmclientes"></local:ViewModelCliente>
    </Window.Resources>
    <Grid Margin="5" Name="gridPrincipal">

        <Grid.RowDefinitions>
            <RowDefinition Height="Auto" />
            <RowDefinition Height="Auto" />
            <RowDefinition Height="auto" />
            <RowDefinition Height="*" />
            <RowDefinition Height="auto" />
        </Grid.RowDefinitions>


        <DockPanel Margin="0,3,30,3" Grid.Row="0"  >
            <Button  Grid.Row="0" Command="local:comandos.voltar"  Width="45"
Height="50" VerticalAlignment="Center" HorizontalAlignment="Left">
                <Image Source="imagens/home.png" ></Image>
            </Button>
            <TextBlock Text="Search by Name" Margin="50,15,0,10"/>
            <TextBox Width="250" Margin="15,10,0,10" HorizontalAlignment="Left"
Text="{Binding Procurar, UpdateSourceTrigger=PropertyChanged}" />

            <Button Content="Guardar PDF"  VerticalAlignment="Center"
HorizontalAlignment="Right" Background="Red" Height="40" Foreground="Black"
Command="local:comandos.pdfTodos"   Width="100"/>

        </DockPanel>
        <Expander Header="Filtros" Grid.Row="1" Background="Black">
            <DockPanel Grid.Row="1" Margin="0,5,0,5">
                <CheckBox Content="Filtar pelo número de livros atribuidos a cada
cliente" Margin="5" IsChecked="{Binding GroupByLimite}"/>
                <CheckBox Content="Filtar por ID" Margin="5" IsChecked="{Binding
SortById}"/>

            </DockPanel>
        </Expander>
        <DockPanel Grid.Row="2" Grid.RowSpan="2"  VerticalAlignment="Stretch"
HorizontalAlignment="Stretch"  >
```

```
            <ListView   Name="lst"   ItemsSource="{Binding ListaClientes}"
Margin="3,10,0,1"   IsSynchronizedWithCurrentItem="True" VerticalAlignment="Stretch"
HorizontalAlignment="Stretch">
                <ListView.GroupStyle>
                    <GroupStyle>
                        <GroupStyle.HeaderTemplate>
                            <DataTemplate>
                                <TextBlock Text="Filtrado por número de livros por
cliente"   FontWeight="Bold"/>
                            </DataTemplate>
                        </GroupStyle.HeaderTemplate>
                    </GroupStyle>
                </ListView.GroupStyle>
                <ListView.View>
                    <GridView>
                        <GridViewColumn Header="ID Cliente" Width="75"
DisplayMemberBinding="{Binding idCliente}" />
                        <GridViewColumn Header="Nome" DisplayMemberBinding="{Binding
nome}" Width="75"/>
                        <GridViewColumn Header="Cartão Cidadão"
DisplayMemberBinding="{Binding cc}" Width="120"></GridViewColumn>
                        <GridViewColumn Header="Limite" DisplayMemberBinding="{Binding
limite}" Width="60"></GridViewColumn>
                    </GridView>
                </ListView.View>
            </ListView>

            <ListView  IsSynchronizedWithCurrentItem="True"

                  ItemsSource="{Binding SelectedItem.registos, ElementName=lst}"
                  x:Name="lstReg" Margin="20,10,0,0"  >

                <ListView.View>

                    <GridView>
                        <GridViewColumn Header="IdRegisto"
DisplayMemberBinding="{Binding idRegisto}"  Width="80"/>
                        <GridViewColumn Header="IdLivro"
DisplayMemberBinding="{Binding idLivro}" Width="80"/>
                        <GridViewColumn Header="Nome Livro"
DisplayMemberBinding="{Binding livro.nomeLivro}" Width="280"/>
                        <GridViewColumn Header="Data Inicial"
DisplayMemberBinding="{Binding dataInicio, StringFormat={}{0:dd/MM/yyyy}}" Width="100"
/>
                        <GridViewColumn Header="Data Máxima"
DisplayMemberBinding="{Binding dataFim,  StringFormat={}{0:dd/MM/yyyy}}" Width="100"/>
                        <GridViewColumn Header="ID Ciente"
DisplayMemberBinding="{Binding idCliente}" Width="80" />
                        <GridViewColumn Header="Data Entrega"
DisplayMemberBinding="{Binding dataEntrega,  StringFormat={}{0:dd/MM/yyyy}}"
Width="100" />

                        <GridViewColumn Header="Estado">
                            <GridViewColumn.CellTemplate>
                                <DataTemplate>
                                    <StackPanel Margin="3,2,6,2">
                                        <Controls:ToggleSwitch OnLabel="Alugado"
OffLabel="Entregue" IsChecked="{Binding entregar }" IsEnabled="{Binding entregar}"
Name="toggleEntrega"/>
                                    </StackPanel>
                                </DataTemplate>
                            </GridViewColumn.CellTemplate>
```

```
                        </GridViewColumn>
                        <GridViewColumn Header="update">
                            <GridViewColumn.CellTemplate>
                                <DataTemplate>
                                    <StackPanel Margin="6,2,6,2">
                                        <Button Style="{StaticResource
AccentedSquareButtonStyle}" Foreground="White" BorderThickness="1"  Content="Update"
Name="updatebtn" Width="70" Background="SteelBlue"
Command="local:comandos.btnEntregarListView"   />
                                    </StackPanel>
                                </DataTemplate>
                            </GridViewColumn.CellTemplate>
                        </GridViewColumn>
                    </GridView>
                </ListView.View>
            </ListView>
        </DockPanel>
        <StackPanel Grid.Row="4" Margin="0,8,0,0" Background="LightGray">

            <Grid >
                <Grid.RowDefinitions>
                    <RowDefinition/>
                    <RowDefinition />
                    <RowDefinition/>
                    <RowDefinition />
                    <RowDefinition Height="auto"/>
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="100"/>
                    <ColumnDefinition />
                    <ColumnDefinition Width="100"/>
                    <ColumnDefinition Width="auto"/>
                </Grid.ColumnDefinitions>

                <TextBlock Margin="5,2,0,0" Text="Cliente Selecionado"
FontWeight="Bold" Grid.ColumnSpan="2" Grid.Row="0"/>
                <TextBlock Text="ID Cliente" Margin="5,8,0,0" Grid.Row="1"
Grid.Column="0"/>
                <TextBlock Name="txtid" Margin="0,9,0,33" Text="{Binding
ClienteCorrente.idCliente,Mode=TwoWay}" Grid.Row="1" Grid.Column="1"
Grid.RowSpan="2"/>

                <TextBlock Text="Nome" Margin="5,8,0,0" Grid.Row="2" Grid.Column="0"/>
                <TextBox  Name="txtnome" Margin="0,8,0,0" Text="{Binding
ClienteCorrente.nome,Mode=TwoWay }" Grid.Column="1" Grid.Row="2"/>

                <TextBlock Text="Livros" Margin="5,8,0,0" Grid.Row="3"
Grid.Column="0"/>
                <ComboBox Name="combLivrosEdit" Margin="0,8,0,0"
SelectedValue="{Binding Path=idLivro}"   SelectedValuePath="idLivro"
DisplayMemberPath="nomeLivro" Grid.Column="1" Grid.Row="3"/>

                <DockPanel Grid.Row="4" Grid.RowSpan="5" Grid.Column="1"
Margin="0,10,0,0" >
                    <StackPanel Orientation="Horizontal">
                        <Button Style="{StaticResource AccentedSquareButtonStyle}"
Foreground="Black" BorderThickness="1" FontWeight="Bold" Content="Apagar Cliente"
Width="100" Height="35" DockPanel.Dock="Right"
Command="local:comandos.btnApagarCliente"  Margin="0,0,5,5"></Button>
```

```
                        <Button Style="{StaticResource AccentedSquareButtonStyle}"
Foreground="Black" BorderThickness="1" FontWeight="Bold" Content="Alugar Livro"
Width="100" Height="35" DockPanel.Dock="Right"  Margin="5,0,5,5"
Command="local:comandos.btnAlugarLivro" ></Button>
                        <Button Style="{StaticResource AccentedSquareButtonStyle}"
Foreground="Black" BorderThickness="1" FontWeight="Bold" Content="Atualizar"
Width="100" Height="35" DockPanel.Dock="Left"  Margin="5,0,5,5"
Command="local:comandos.btnAtualizaCliente"  ></Button >

                    </StackPanel>
                </DockPanel>
                <Button  Style="{DynamicResource MetroCircleButtonStyle}"
Grid.Row="1" Grid.Column="3" Grid.RowSpan="5" Margin="0,0,20,0"
HorizontalAlignment="Center" VerticalAlignment="Center" Background="LightGray"
BorderBrush="LightGray" Command="local:comandos.botaoCriarMudar" Width="100">

                    <StackPanel>
                        <Image Source="imagens/users.png" />
                    </StackPanel>
                </Button>
            </Grid>
        </StackPanel>
    </Grid>
</Controls:MetroWindow>
```

Código em Code-Behind:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using MahApps.Metro.Controls;
using MahApps.Metro.Controls.Dialogs;

namespace projetoMVVM
{
    public partial class criarClientes : MetroWindow
    {
         public ViewModelCliente vmclientes { get; set; }

        public criarClientes()
        {
            InitializeComponent();
            //associa o xaml e o DataContext à vmclientes
            vmclientes = (ViewModelCliente)Resources["vmclientes"];
            vmclientes.onPropertyChanged("ListaClientes");
            this.DataContext = vmclientes;
            //Evento que é disparado quando se fecha a janela
            vmclientes.ClosingRequest += (sender, e) => this.Close();
            //prenche combobox
            getCombo();
```

```
            //Binding dos comandos e do métodos com os botões (xaml)
            CommandBindings.Add(new CommandBinding(
              comandos.botaoCriarMudar,
              (sender, e) => vmclientes.criarCliView(),
               (sender, e) => e.CanExecute = true  ));

            CommandBindings.Add(new CommandBinding(
              comandos.btnEntregarListView,
              (sender, e) => vmclientes.entregar(lstReg),
               (sender, e) => e.CanExecute = true ));

            CommandBindings.Add(new CommandBinding(
       comandos.btnAtualizaCliente,
       (sender, e) => vmclientes.atualizaCliente(vmclientes.ClienteCorrente, txtnome),
       (sender, e) => e.CanExecute = true ));

            CommandBindings.Add(new CommandBinding(
      comandos.btnAlugarLivro,
      (sender, e) => vmclientes.alugarLivro(vmclientes.ClienteCorrente,
combLivrosEdit, lstReg),
      (sender, e) => e.CanExecute = true ));

            CommandBindings.Add(new CommandBinding(
          comandos.btnApagarCliente,
          (sender, e) => vmclientes.apagarCliente(vmclientes.ClienteCorrente),
           (sender, e) => e.CanExecute = vmclientes.check() ));

            CommandBindings.Add(new CommandBinding(
          comandos.voltar,
          (sender, e) => vmclientes.voltarHome(),
           (sender, e) => e.CanExecute = true));


            CommandBindings.Add(new CommandBinding(
          comandos.pdfTodos,
          (sender, e) => vmclientes.pdfTodo(),
           (sender, e) => e.CanExecute = true));

        }

        public void getCombo()
            //Prenche a ComboBox com uma query feita em LINQ  com os livros que podem
ser alugados
        {
            using (biblioProjetoMVVMEntities biblio = new biblioProjetoMVVMEntities())
            {
                var livrosDados = (from liv in biblio.livros
                                   where liv.estado == false
                                   select new
                                   {
                                       nomeLivro = liv.nomeLivro,
                                       idLivro = liv.idLivro,
                                   }).ToList();
                combLivrosEdit.ItemsSource = livrosDados;
            }
        }


    }
}
```

Anexo 8 – CriarCli.xaml

Código em XAML:

```
<Controls:MetroWindow x:Class="projetoMVVM.criarCli"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:projetoMVVM"
        xmlns:Controls="clr-namespace:MahApps.Metro.Controls;assembly=MahApps.Metro"
        xmlns:Dialog="clr-
namespace:MahApps.Metro.Controls.Dialogs;assembly=MahApps.Metro"
        mc:Ignorable="d"
        Title="criarCli" Height="490" Width="500">
    <Window.Resources>
        <local:ViewModelCliente x:Key="nomes"></local:ViewModelCliente>
    </Window.Resources>
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="*"></RowDefinition>
            <RowDefinition Height="*"></RowDefinition>
            <RowDefinition Height="auto"></RowDefinition>
        </Grid.RowDefinitions>
        <Image Grid.Row="0" Source="imagens/adduser.png" Margin="20"></Image>
        <StackPanel Grid.Row="1" Orientation="Vertical">

            <TextBox Controls:TextBoxHelper.Watermark="Nome" Margin="10"
HorizontalAlignment="Center" Height="23" Name="txtCli"  VerticalAlignment="Center"
Width="120"/>
            <TextBox Controls:TextBoxHelper.Watermark="NIF"
HorizontalAlignment="Center" Height="23"  Margin="10" Name="txtNif"
VerticalAlignment="Top" Width="120"/>
            <ComboBox  VerticalAlignment="Center" Width="280" Margin="10"
HorizontalAlignment="Center" Name="combLivros"  SelectedValue="{Binding Path=idLivro}"
SelectedValuePath="idLivro" DisplayMemberPath="nomeLivro" Height="26"/>


            <Controls:ToggleSwitch OnLabel="Alugado" OffLabel="No" IsChecked="True"
Width="150" Margin="5"  Name="toggSwitc" HorizontalAlignment="Center"/>
        </StackPanel>
        <StackPanel Grid.Row="2" Orientation="Horizontal" VerticalAlignment="Center"
Margin="10" HorizontalAlignment="Center"  >
            <Button Style="{StaticResource AccentedSquareButtonStyle}"
Foreground="Black" BorderThickness="1" FontWeight="Bold" Width="150"
Name="btnAtualizar"  Height="40"  Content="Atualizar"
Command="local:comandos.insertCliente" Margin="0,0,30,0" ></Button>
            <Button Style="{StaticResource AccentedSquareButtonStyle}"
Foreground="Black" BorderThickness="1" FontWeight="Bold" Width="150"
Name="btnCancelar" Height="40" Margin="10" Content="Cancelar"
Command="local:comandos.btnCancelarCliente" ></Button>
        </StackPanel>
    </Grid>
</Controls:MetroWindow>
```

Código em Code-Behind:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using MahApps.Metro.Controls;
using MahApps.Metro.Controls.Dialogs;

namespace projetoMVVM
{
    public partial class criarCli : MetroWindow
    {
        public ViewModelCliente vmclientes { get; set; }

        public criarCli()
        {
            InitializeComponent();
            //associa o xaml e o DataContext à vmclientes
            vmclientes = (ViewModelCliente)Resources["nomes"];
            this.DataContext = vmclientes;
            //Evento que é disparado quando se fecha a janela
            vmclientes.ClosingRequest += (sender, e) => this.Close();
            getCombo();

            //Binding do comando e do método com o botão (xaml)
            CommandBindings.Add(new CommandBinding(
              comandos.insertCliente,
              (sender, e) => vmclientes.insertPessoa(txtCli, txtNif,  toggSwitc,
combLivros),
               (sender, e) => e.CanExecute = true ));

            CommandBindings.Add(new CommandBinding(
         comandos.btnCancelarCliente,
         (sender, e) => vmclientes.cancelarBtnCliente(),
          (sender, e) => e.CanExecute = true));
        }
        private void getCombo()
        { //Prenche a ComboBox com uma query feita em LINQ  com os livros que podem
ser alugados
            using (biblioProjetoMVVMEntities biblio = new biblioProjetoMVVMEntities())
            {
                var livrosDados = (from liv in biblio.livros where liv.estado == false
                                   select new
                                   {
                                       nomeLivro = liv.nomeLivro,
                                       idLivro = liv.idLivro,
                                   }).ToList();
                combLivros.ItemsSource = livrosDados;
            }
        }     }  }
```

Anexo 9 – catalogo.xaml

Código em XAML:

```
<Controls:MetroWindow  x:Class="projetoMVVM.catalogo"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
         xmlns:Controls="clr-namespace:MahApps.Metro.Controls;assembly=MahApps.Metro"
        xmlns:Dialog="clr-
namespace:MahApps.Metro.Controls.Dialogs;assembly=MahApps.Metro"
        xmlns:local="clr-namespace:projetoMVVM"
        mc:Ignorable="d"
        Title="catalogo" Height="650" Width="900"
WindowStartupLocation="CenterScreen">
    <Window.Resources>
        <local:arrayimagem x:Key="conversor"></local:arrayimagem>
        <local:ViewModelLivros x:Key="vmlivros"></local:ViewModelLivros>

    </Window.Resources>
    <Grid  DataContext="{Binding Source={StaticResource vmlivros}}">
        <Grid.RowDefinitions>
            <RowDefinition Height="*"></RowDefinition>
            <RowDefinition Height="143"/>
            <RowDefinition Height="157"/>
            <RowDefinition Height="235"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="auto"></ColumnDefinition>
            <ColumnDefinition Width="*"></ColumnDefinition>
        </Grid.ColumnDefinitions>

        <Button  Grid.Row="0" Command="local:comandos.voltar" VerticalAlignment="Top"
Width="45" Height="50"  HorizontalAlignment="Left">
            <StackPanel >
                <Image Source="imagens/home.png" ></Image>
            </StackPanel>
        </Button>

        <ListView Grid.Row="0" Grid.RowSpan="3" Name="lst" ItemsSource="{Binding
ListaLivros}" Margin="10,55,0,5"  IsSynchronizedWithCurrentItem="True"
VerticalAlignment="Stretch"  HorizontalAlignment="Stretch" >
            <ListView.View>
                <GridView>
                    <GridViewColumn Header="ID" DisplayMemberBinding="{Binding
idLivro}" Width="70"/>
                    <GridViewColumn Header="Nome Livro" DisplayMemberBinding="{Binding
nomeLivro}" Width="270"></GridViewColumn>
                    <GridViewColumn Header="Autor" DisplayMemberBinding="{Binding
nomeAutor}" Width="140"></GridViewColumn>
                    <GridViewColumn Header="Editora" DisplayMemberBinding="{Binding
editora}" Width="100"></GridViewColumn>
                    <GridViewColumn Header="Alugado" DisplayMemberBinding="{Binding
estado}" Width="100"></GridViewColumn>
                </GridView>

            </ListView.View>
        </ListView>
```

```
        <Image Name="imgLivros" Margin="10" Source="{Binding LivroCorrente.foto,
Mode=TwoWay, Converter={StaticResource conversor}}"  Grid.Column="1" Grid.Row="0"
Grid.RowSpan="3" HorizontalAlignment="Right" ></Image>

        <StackPanel Grid.Row="3"  Orientation="Horizontal" VerticalAlignment="Center"
Margin="10,5,5,0" Background="LightBlue" >
            <StackPanel Orientation="Vertical" >
                <TextBlock Text="Nome Livro" Width="100" FontWeight="Bold"
></TextBlock>
                <TextBox Name="txtNomeCriar" Text="{Binding
LivroCorrente.nomeLivro,Mode=TwoWay }" Margin="10" Width="180" Height="30" ></TextBox>
                <TextBlock Text="Nome Autor" Width="100" FontWeight="Bold"
></TextBlock>
                <TextBox Name="txtAutorCriar" Text="{Binding
LivroCorrente.nomeAutor,Mode=TwoWay }" Width="180" Margin="10" Height="30" ></TextBox>
                <TextBlock Text="Nome Editora"  Width="100" FontWeight="Bold"
></TextBlock>
                <TextBox Name="txtEditoraCriar" Text="{Binding
LivroCorrente.editora,Mode=TwoWay }" Margin="8,8,8,10" Width="180" Height="30"
></TextBox>

            </StackPanel>
            <StackPanel Orientation="Vertical" VerticalAlignment="Center"
Margin="80,10,10,10">
                <Button Style="{StaticResource AccentedSquareButtonStyle}"
Foreground="Black" BorderThickness="2" FontWeight="Bold" Content="Atualizar Livro"
Width="130" Margin="10"  Command="local:comandos.btnAtualizaLivro"  Height="40"   />
                <Button  Style="{StaticResource AccentedSquareButtonStyle}"
Foreground="Black" BorderThickness="2" FontWeight="Bold" Content="Load Foto"
Width="130"  Height="40" Command="local:comandos.foto_click"  />
                <Button  Style="{StaticResource AccentedSquareButtonStyle}"
Foreground="Black" BorderThickness="2" FontWeight="Bold" Content="Apagar Livro"
Width="130"  Margin="10" Height="40" Command="local:comandos.btnapagarLivro"  />
                <Button  Style="{StaticResource AccentedSquareButtonStyle}"
Foreground="Black" Background="Red" BorderThickness="2" FontWeight="Bold"
Content="Obter PDF"  Width="130"  Margin="10" Height="40"
Command="local:comandos.btnPDF"  />

            </StackPanel>
        </StackPanel>
        <Button  Background="CornflowerBlue"   Margin="5,15,10,30"
Command="local:comandos.btnInserirLivroView" VerticalAlignment="Center"
HorizontalAlignment="Right" Grid.Column="1" Grid.Row="3" Width="150" >
            <StackPanel>
                <TextBlock Text="Novo Livro" TextAlignment="Center"
Foreground="White" FontWeight="Bold" Width="180"  HorizontalAlignment="Right"
FontSize="15"  Padding="4"></TextBlock>
                <Image Source="imagens/books.png"  ></Image>
            </StackPanel>
        </Button>
    </Grid>
</Controls:MetroWindow>
```

Código em Code-Behind:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Globalization;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using MahApps.Metro.Controls;
using MahApps.Metro.Controls.Dialogs;
using System.Windows.Navigation;


namespace projetoMVVM
{
    public partial class catalogo : MetroWindow
    {
        ViewModelLivros vmLivros { get; set; }
        public catalogo()
        {
            InitializeComponent();
            //associa o xaml e o DataContext à vmLivros
            vmLivros = (ViewModelLivros)Resources["vmlivros"];
            this.DataContext = vmLivros;
            //Evento que é disparado quando se fecha a janela
            vmLivros.ClosingRequest += (sender, e) => this.Close();

            //Binding dos comandos e do métodos com os botões (xaml)
            CommandBindings.Add(new CommandBinding(
            comandos.foto_click,
            (sender, e) => vmLivros.foto_click(this.imgLivros),
             (sender, e) => e.CanExecute = true));

            CommandBindings.Add(new CommandBinding(
            comandos.btnInserirLivroView,
            (sender, e) => vmLivros.viewCriarLivro(),
             (sender, e) => e.CanExecute = true));

            CommandBindings.Add(new CommandBinding(
          comandos.btnAtualizaLivro,
          (sender, e) => vmLivros.atualizaLivro(vmLivros.LivroCorrente, txtNomeCriar,
txtAutorCriar, txtEditoraCriar),
           (sender, e) => e.CanExecute = true));

            CommandBindings.Add(new CommandBinding(
          comandos.btnapagarLivro,
          (sender, e) => vmLivros.apagarLivro(vmLivros.LivroCorrente),
           (sender, e) => e.CanExecute = vmLivros.checkAdmin() ));
```

```
            CommandBindings.Add(new CommandBinding(
            comandos.voltar,
            (sender, e) => vmLivros.voltarHome() ,
             (sender, e) => e.CanExecute = true));

            CommandBindings.Add(new CommandBinding(
            comandos.btnPDF,
            (sender, e) => vmLivros.pdfLivro(),
             (sender, e) => e.CanExecute = true));
        }
    }

    //Conversor de imagem
    public class arrayimagem : IValueConverter
    {
        public object Convert(object value, Type targetType, object parameter,
CultureInfo culture)
        {
            byte[] fotoarray = (byte[])value;
            BitmapImage foto = utils.ConvertByteArrayToBitmapImage(fotoarray);
            return foto;
        }
        public object ConvertBack(object value, Type targetType, object parameter,
CultureInfo culture)
        {
            byte[] fotoarray;
            BitmapImage foto = (BitmapImage)value;
            fotoarray = utils.ConvertBitmapSourceToByteArray(foto);
            return fotoarray;
        }
    }

}
```

Anexo 9 – criarLivro.xaml

Código em XAML:

```
<Controls:MetroWindow x:Class="projetoMVVM.criarLivro"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:projetoMVVM"
        xmlns:Controls="clr-namespace:MahApps.Metro.Controls;assembly=MahApps.Metro"
        xmlns:Dialog="clr-
namespace:MahApps.Metro.Controls.Dialogs;assembly=MahApps.Metro"
        mc:Ignorable="d"
        Title="criarLivro" Height="450" Width="800">
    <Window.Resources>

        <local:ViewModelLivros x:Key="vmlivros"></local:ViewModelLivros>
    </Window.Resources>
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="250"></ColumnDefinition>
            <ColumnDefinition Width="*"></ColumnDefinition>
        </Grid.ColumnDefinitions>
        <StackPanel Grid.Row="3" Grid.ColumnSpan="2" Orientation="Vertical"
VerticalAlignment="Center" HorizontalAlignment="Center">
            <TextBox Controls:TextBoxHelper.Watermark="Nome Livro"
Name="txtNomeCriar" Margin="8" Width="200" Height="40" ></TextBox>
            <TextBox Controls:TextBoxHelper.Watermark="Nome Autor"
Name="txtAutorCriar" Margin="8" Width="200" Height="40" ></TextBox>
            <TextBox Controls:TextBoxHelper.Watermark="Nome Editora"
Name="txtEditoraCriar" Width="200" Margin="8" Height="40" ></TextBox>

            <StackPanel Orientation="Horizontal" VerticalAlignment="Center"
HorizontalAlignment="Center">
                <Button  Style="{StaticResource AccentedSquareButtonStyle}"
Foreground="Black" BorderThickness="1" FontWeight="Bold" Width="150" Height="50"
Margin="10" Content="Adicionar Livro"
Command="local:comandos.btnInserirLivro"></Button>
                <Button Style="{StaticResource AccentedSquareButtonStyle}"
Foreground="Black" BorderThickness="1" FontWeight="Bold" Content="Load Foto"
Height="50" Width="80"  Command="local:comandos.foto_click"  Grid.Column="3"
Grid.Row="5"></Button>
                <Button Style="{StaticResource AccentedSquareButtonStyle}"
Foreground="Black" BorderThickness="1" FontWeight="Bold" Content="Cancelar"
Height="50" Width="80" Command="local:comandos.cancelarLivro"  Grid.Column="3"
Margin="10" Grid.Row="5"></Button>
            </StackPanel>
        </StackPanel>
        <Image x:Name="imgLivros" Grid.Column="0" VerticalAlignment="Stretch"
HorizontalAlignment="Stretch" Margin="35" />

    </Grid>


</Controls:MetroWindow>
```

Código em Code-Behind:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using MahApps.Metro.Controls;
using MahApps.Metro.Controls.Dialogs;

namespace projetoMVVM
{
    public partial class criarLivro : MetroWindow
    {
        ViewModelLivros vmLivros { get; set; }

        public criarLivro()
        {
            InitializeComponent();
            //associa o xaml e o DataContext à vmLivros
            vmLivros = (ViewModelLivros)Resources["vmlivros"];
            this.DataContext = vmLivros;
            //Evento que é disparado quando se fecha a janela
            vmLivros.ClosingRequest += (sender, e) => this.Close();

            //Binding dos comandos e do métodos com os botões (xaml)
            CommandBindings.Add(new CommandBinding(
           comandos.foto_click,
           (sender, e) => vmLivros.foto_click(this.imgLivros),
            (sender, e) => e.CanExecute = true));

            CommandBindings.Add(new CommandBinding(
            comandos.btnInserirLivro,
            (sender, e) => vmLivros.insertLivro(txtNomeCriar, txtAutorCriar,
txtEditoraCriar, imgLivros),
             (sender, e) => e.CanExecute = true));

            CommandBindings.Add(new CommandBinding(
             comandos.cancelarLivro,
            (sender, e) => vmLivros.cancelLivro(),
            (sender, e) => e.CanExecute = true));
        }
    }
}
```